DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Comparison of Selected Model Order Reduction methods

Studienarbeit (T3_3101)

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Florian Braun

Januar 2018

-Sperrvermerk-

Abgabedatum:	22. Mai 2023
Bearbeitungszeitraum:	14.10.2022 - 22.05.2023
Matrikelnummer, Kurs:	7433149, TINF20B1
Gutachter der Dualen Hochschule:	Lutz Gröll

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit (T3_3101) mit dem Thema:

Comparison of Selected Model Order Reduction methods

gemäß § 5 der "Studien- und Prüfungsordnung DHBW Technik" vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den May 19, 2023

Nachname, Vorname

Abstract

- Deutsch -

Dies ist der Beginn des Abstracts. Für die finale Bachelorarbeit musst du ein Abstract in deinem Dokument mit einbauen. So, schreibe es am besten jetzt in Deutsch und Englisch. Das Abstract ist eine kurze Zusammenfassung mit ca. 200 bis 250 Wörtern.

Versuche in das Abstract folgende Punkte aufzunehmen: Fragestellung der Arbeit, methodische Vorgehensweise oder die Hauptergebnisse deiner Arbeit.

Contents

List of abbreviations

DE	Differential Equation
DFT	Discrete Fourier Transform
FEM	Finite Element Method
FT	Fourier Transform
FFT	Fast Fourier Transform
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
LS	Least Squares
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation

List of Figures

List of Tables

source-codeverzeichnis

1. Introduction

With the development of semiconductor based computers mankind has access to the most powerful computing machines ever created by humans. Still those cutting edge technologies often do not provide enough computational power to solve problems in maths, physics or engineering. Often those problems come in form of partial differential equations, to come up with approximate solutions discretization is used to transform PDEs into systems of ordinary differential equations. This resulting system of ODEs is often still too large to use them for computations directly. Methods to reduce the size of those systems are called model order reduction methods.

1.1. Problem statement

Based on the problem and its setting a variety of model reduction methods can be chosen. Each of them approximate the original system in some optimal manner. The goal of this paper is to compare the properties of a defined set of model order reduction methods. This comparison consists of a comparison in the error of the reduced order models and the time it takes to compute them. The system that gets reduced describes the heat transfer in a piece of wire that can be heated or cooled arbitrarily.

1.2. Outline

The fundamentals are covered first. It begins with the basics about heat equation and how the already mentioned heating or cooling is done respectively. Also some naive method to obtain an approximate solution to it is covered. However the next part of that chapter will discuss the finite element method since this technique is more powerful than the previously stated method. After the singular value decomposition is covered since it sees a lot of usage in the model order reduction methods. Since most of the methods come from control theory some basic control theory is covered in the subsequent section. Chapter three covers the workings of the model order reduction methods. It begins with the proper orthogonal decomposition. Balanced truncation, modal truncation and hankel norm approximation follows. Last but not least it gets demonstrated that modal truncation equals balanced truncation for this system. In chapter four the implementation of those methods and the implementation of the finite element solver is discussed. After that numerical experiments are conducted. Here the time domain error, frequency domain error and the time it takes to compute the reduced order models are measured and compared.

2. Fundamentals

This chapter covers the fundamentals, necessary for understanding the model order reduction methods. First of all a system to apply those methods to is needed. Here the 1D heat equation was chosen, since it is fairly easy to solve and to understand the results. After that the method of finite elements is introduced. This method is used to discretize the 1D heat equation into a solvable system of ODEs since it can handle boundary conditions and initial values. One of the most important fundamentals of this paper is the singular value decomposition, since it takes heavy use in the model order reduction methods covered. The last Section of this chapter will cover some control system fundamentals. Those are needed since three of the model order reduction methods are used in a control system context, hence some explanation on the terminology is needed.

2.1. Heat Equation

The conduction of heat within a medium can be described using the following partial differential equation (PDE):

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \,. \tag{2.1}$$

Here u is a function of space and time and α is a positive constant. For this paper u will be defined in terms of one spatial dimension [1]

$$u := u(x,t) \tag{2.2}$$

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{2.3}$$

$$x \in \chi \subset \mathbb{R} \quad t \in \tau \subset \mathbb{R} \tag{2.4}$$

$$x_0 \leq x \leq x_n \quad t_0 \leq t \leq t_n \,. \tag{2.5}$$

.

In order to not only model the conduction of heat within a medium but also a heating process, a new function $h : \chi \times \tau \to \mathbb{R}$ is introduced:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + h(x,t) \,. \tag{2.6}$$

For this paper it is assumed that the initial condition is known:

$$f : \chi \to \mathbb{R} \tag{2.7}$$
$$u(x, t_0) = f(x)\,. \tag{2.8}$$

Applying the Fourier transform (FT) w.r.t x to 2.6 yields the inhomogeneous ordinary differential equation (ODE):

$$\hat{u} = \mathfrak{F}(u) \quad \hat{h} = \mathfrak{F}(h) \tag{2.9}$$
$$\frac{d}{dt}\hat{u} = -\alpha\omega^2\hat{u} + \hat{h}\,. \tag{2.10}$$

A solution to 2.10 is given by:

$$\hat{u} = \hat{u}_0 + \hat{u}_p\,. \tag{2.11}$$

Where $\hat{u}_0$ is the homogeneous solution and $\hat{u}_p$ is the particular integral. In order to solve this ODE for the particular integral $\hat{h}$ has to be known [2]. The choice of h is, except for some restrictions, arbitrary. Therefore an approximate solution to 2.10 $\hat{u}_a$ is obtained by the forward euler scheme:

$$\frac{d}{dt}\hat{u} \approx \frac{\Delta\hat{u}}{\Delta t} \tag{2.12}$$
$$\hat{u}_{t+1} = \hat{u}_t + \Delta t(-\alpha\omega^2\hat{u} + \hat{h}) \tag{2.13}$$
$$\hat{u}_a = \begin{bmatrix}\hat{u}_{t_0} & \cdots & \hat{u}_{t_n}\end{bmatrix}\,. \tag{2.14}$$

In order to apply the euler scheme successfully an initial condition $\hat{u}_0$ has to be known. This initial condition is obtained by applying the discrete Fourier transform (DFT) to an initial temperature distribution along x:

$$\hat{u}_0 = \mathfrak{F}\{f(x)\}\,. \tag{2.15}$$

[3]
The forward euler scheme is used here because it is fairly easy to implement. By applying the inverse discrete Fourier transform (IDFT) to $\hat{u}_a$ an approximate solution to 2.6 can be obtained. To decrease computing time, the fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT) is used instead of the DFT and IDFT. A problem with solving the heat equation in this way is that it cannot handle boundary conditions. In order to solve the heat equation as initial boundary value problem (IBVP) the finite element method is used.

2.2. Finite Element Method

The finite element method (FEM) is a method to approximate solutions for differential equations (DE) within a certain domain Ω. This is done by discretizing the spatial domain. Assume that a DE is given by

$$m, n \in \mathbb{N} \quad \zeta \in \Omega \subset \mathbb{R} \quad m \geq 1 \tag{2.16}$$

$$\frac{\partial^m y}{\partial \zeta^m} - g(y) = r(\zeta, t)\,. \tag{2.17}$$

It is assumed that g is a linear function that can also contain partial derivatives of y w.r.t. time, y takes the value 0 at the boundary Γ and $y(\zeta, 0) = f(\zeta)$. An approximate solution to y is given by μ, which is expressed as a sum of basis functions contained in the set ϕ:

$$\mu(\zeta, t) = \sum_{j=1}^{N} c_j(t)\phi_j(\zeta)\,. \tag{2.18}$$

The residual is defined as

$$\mathfrak{r} = \frac{\partial^m \mu}{\partial \zeta^m} - g(\mu) - r(\zeta, t)\,. \tag{2.19}$$

Furthermore the residual is required to be orthogonal to all basis functions

$$\langle \mathfrak{r}, \phi_k \rangle = 0 \quad \forall \phi_k \in \phi\,. \tag{2.20}$$

Since the functions in ϕ are known, it is only required to find the coefficients $c_j(t)$ in 2.18. To find those coefficients 2.20 needs to be expressed as follows

$$\int_\Omega \frac{\partial^m \mu}{\partial \zeta^m}\phi_k \, d\zeta - \int_\Omega g(\mu)\phi_k \, d\zeta = \int_\Omega r(\zeta, t)\phi_k \, d\zeta \quad \forall \phi_k \in \phi\,. \tag{2.21}$$

If μ is substituted with 2.18 the following is obtained

$$\sum_{j=1}^{N} \left(\left(\int_\Omega \frac{\partial^m \phi_j}{\partial \zeta^m}\phi_k \, d\zeta \right) c_j(t) - g\left(\left(\int_\Omega \phi_k \phi_j d\zeta \right) c_j(t) \right) \right) = \int_\Omega r(\zeta, t)\phi_k \, d\zeta \quad \forall \phi_k \in \phi\,. \tag{2.22}$$

It is also necessary to apply the divergence theorem to the first integral term taking into account that y at Γ is 0. Since ζ is one dimensional, the divergence theorem becomes integration by parts

$$\int_\Omega \frac{\partial^m \phi_j}{\partial \zeta^m}\phi_k \, d\zeta = -\int_\Omega \frac{\partial^{m-1} \phi_j}{\partial \zeta^{m-1}} \frac{\partial \phi_k}{\partial \zeta} \, d\zeta \quad \forall \phi_k \in \phi\,. \tag{2.23}$$

Combining 2.22 and 2.23 yields

$$-\sum_{j=1}^{N}\left(\left(\int_{\Omega}\frac{\partial^{m-1}\phi_j}{\partial\zeta^{m-1}}\frac{\partial\phi_k}{\partial\zeta}\,d\zeta\right)c_j(t)+g\left(\left(\int_{\Omega}\phi_k\phi_j d\zeta\right)c_j(t)\right)\right)=\int_{\Omega}r(\zeta,t)\phi_k\,d\zeta\quad\forall\phi_k\in\phi\,. \tag{2.24}$$

This formulation leads to a system of ODEs or a system of linear equations that can be solved either analytically or numerically.

2.2.1. Solving the Heat Equation using FEM

This formulation of FEM can be applied to 2.6

$$\Omega=\chi\quad\Gamma=\{x_0,x_n\} \tag{2.25}$$

$$y(\zeta,t)=-u(x,t)\quad g(u)=-\frac{1}{\alpha}\frac{\partial u}{\partial t} \tag{2.26}$$

$$m=2\quad r(\zeta,t)=\frac{1}{\alpha}h(x,t) \tag{2.27}$$

$$u(x,0)=f(x)\quad u(x_0,t)=0\quad u(x_n,t)=0\,. \tag{2.28}$$

The set of basis functions is defined as a set of piecewise linear functions with constant step size Δx

$$\phi_j(x)=\begin{cases}(x-x_{j-1})/\Delta x), & x_{j-1}\le x<x_j\\ (x_{j+1}-x)/\Delta x), & x_j\le x<x_{j+1}\\ 0, & \text{otherwise}\end{cases}\quad. \tag{2.29}$$

[4]
The step size Δx is defined by $\Delta x=\frac{x_n-x_0}{n-1}$. This results in the following system of ODEs

$$\sum_{j=1}^{N}\left(\int_{\chi}\phi_j\phi_k dx\right)\frac{dc_j}{dt}=\alpha\sum_{j=1}^{N}\left(-\int_{\chi}\frac{d\phi_j}{dx}\frac{d\phi_j}{dx}dx\right)c_j(t)+\int_{\chi}h(x,t)\phi_k dx\quad\forall\phi_k\in\phi\,. \tag{2.30}$$

Since $\phi_k\phi_j\neq 0$ for $k=j,k=j\pm1$ (2.30) can be expressed as

$$q_1\dot{c}_{j-1}+q_2\dot{c}_j+q_1\dot{c}_{j+1}=p_1c_{j-1}+p_2c_j+p_1c_{j+1}+H(x,t)\,. \tag{2.31}$$

where p and q are some constants derived later. For $j=1$ and $j=n$ some new coefficients c_0,c_{n+1} are introduced. To force the boundary condition they are set to zero $c_0=c_{n+1}=0$. It follows that $h(x,t)$ and $c_j(0)$ have to satisfy the boundary condition too.

Using matrix notation this becomes

$$M^{N\times N}, K^{N\times N} \tag{2.32}$$

$$M\dot{c} = Kc + d\,. \tag{2.33}$$

The matrices M and K can be easily computed (Appendix A.1)

$$m_{ij} = \begin{cases} \frac{2\Delta x}{3}, & k = j \\ \frac{\Delta x}{6}, & |k-j| = 1 \\ 0, & otherwise \end{cases} \qquad k_{ij} = \begin{cases} \frac{-2\alpha}{\Delta x}, & k = j \\ \frac{\alpha}{\Delta x}, & |k-j| = 1 \\ 0, otherwise \end{cases} \quad . \tag{2.34}$$

Furthermore to solve this system of ODEs numerically an initial condition c_0 has to be known [3]. It can be obtained using a least squares (LS) approach [5]

$$\sum_{j=1}^{N} \langle \phi_j, \phi_k \rangle c_j(0) = \langle f, \phi_k \rangle \quad \forall \phi_k \in \phi \tag{2.35}$$

$$Mc_0 = F \tag{2.36}$$

$$c_0 = M^{-1}F\,. \tag{2.37}$$

However since the basis functions are piecewise linear functions, the coefficients are $c_0 = \left[f(0), f(\Delta x), ..., f(n)\right]^T$ **[Gustafsson2011g]**. It is necessary to approximate d for each point in time using numerical integration schemes. By assuming that the following decomposition holds

$$h(x,t) = \sum_{i=1}^{N} \phi_i(x)\upsilon(t)\,. \tag{2.38}$$

the vector d in 2.33 can be expressed as $M\upsilon(t)$. Observe that by multiplying 2.33 with M^{-1} (A.2) yields a system of ODEs

$$\dot{c} = M^{-1}Kc + M^{-1}M\upsilon\,. \tag{2.39}$$

This system of ODEs can be solved using an euler scheme

$$c_{t+1} = \Delta t(M^{-1}Kc_t + I\upsilon) + c_t\,. \tag{2.40}$$

Using 2.18 and the computed coefficients c the function $u(x,t)$ can be approximated. However this is equivalent to linear interpolation between c_n and c_{n+1} (Appendix A.3).

2.3. Singular Value Decomposition

The Singular Value Decomposition (SVD) is a matrix factorization with guaranteed existence. It can be used to obtain low rank approximations of a matrix or pseudo inverses for ill posed linear system of equations. It is also related to FT by providing a data specific set of orthogonal bases instead of a generic set of sines and cosines. For this paper the SVD will be used for generating low rank approximations of matrices [6].

2.3.1. Properties

A matrix $X \in \mathbb{R}^{n \times m}$ can be decomposed in the following way

$$X = U\Sigma V^T . \tag{2.41}$$

Here $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{m \times m}$ are unitary matrices and $\Sigma \in \mathbb{R}^{n \times m}$ is a real valued ordered diagonal matrix. The columns of U provide a set of orthonormal basis vectors for the column space of X, V contains orthonormal basis vectors for the row space of X. The matrix Σ assigns a magnitude ('importance') to the product of U and V^T [7]. Since U and V are unitary they have the following property [8]

$$U^T U = UU^T = I \tag{2.42}$$
$$V^T V = VV^T = I . \tag{2.43}$$

In case $n \geq m$ the so called economy SVD can be used to factorize the matrix X

$$X = \begin{bmatrix} \hat{U} & \hat{U}^{\perp} \end{bmatrix} \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} V^T = \hat{U}\hat{\Sigma}V^T . \tag{2.44}$$

The economy SVD omits rows only containing zeros in Σ and the according columns of U. Therefore the dimensionality of $\hat{U}$ and $\hat{\Sigma}$ is less or equal to the dimensionality of U and Σ [6].

2.3.2. Hierarchy of correlations

As already stated, the matrix Σ assigns a magnitude to UV^T. This magnitude can be seen as the square of the variance the bases in U and V capture. Assume that $X^T X$ and XX^T denote correlation matrices [6]. A correlation matrix is a matrix that stores correlation coefficients between multiple measurements. If X has the following properties $X^T X$ and XX^T are correlation matrices.

1.) The column vectors of X have to be zero mean

$$X = \begin{bmatrix} x_1 & \cdots & x_m \end{bmatrix} \tag{2.45}$$

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij} = \mu_i = 0 \quad , 1 \leq i \leq m\,. \tag{2.46}$$

2.) The column vectors of X have to be normalized

$$\sqrt{\sum_{i=1}^{n} x_{ij}^2} = \bar{x_i} = 1 \quad , 1 \leq i \leq m\,. \tag{2.47}$$

The correlation coefficient between two column vectors of X is calculated as follows [9]

$$\mathrm{corr}(x_i, x_{i'}) \frac{(\sum_{j=1}^{n} x_{ij} - \bar{x}_i)(\sum_{j=1}^{n} x_{i'j} - \bar{x}_{i'})}{(\sum_{j=1}^{n}(x_{ij} - \bar{x_i})^2)^{\frac{1}{2}}(\sum_{j=1}^{n}(x_{i'j} - \bar{x}_{i'})^2)^{\frac{1}{2}}}\,. \tag{2.48}$$

Since all column vectors have zero mean and are normalized this becomes [10]

$$\mathrm{corr}(x_i, x_{i'}) = cov(x_i, x_{i'}) = x_i^T x_{i'}\,. \tag{2.49}$$

This resembles the entries of XX^T and X^TX. A vector e that maximizes the variance of the projection of x_i onto e with the restriction $\|e\| = 1$ are the eigenvectors of the according correlation matrix. The eigenvalue of e denoted as λ is equivalent to the variance of x_i projected onto e [11].
Since X can be de-constructed using the SVD, XX^T and X^TX are equal to

$$XX^T = U\begin{bmatrix}\hat{\Sigma}\\ 0\end{bmatrix} V^TV \begin{bmatrix}\hat{\Sigma} & 0\end{bmatrix} U^T = U\begin{bmatrix}\hat{\Sigma}^2 & 0\\ 0 & 0\end{bmatrix} U^T \tag{2.50}$$

$$X^TX = V\begin{bmatrix}\hat{\Sigma} & 0\end{bmatrix} U^TU \begin{bmatrix}\hat{\Sigma}\\ 0\end{bmatrix} V^T = V\hat{\Sigma}^2V^T\,. \tag{2.51}$$

By multiplying U and V respectively on the right side 2.50 and 2.51 become

$$XX^TU = U\begin{bmatrix}\hat{\Sigma}^2 & 0\\ 0 & 0\end{bmatrix} \tag{2.52}$$

$$X^TXV = V\hat{\Sigma}^2\,. \tag{2.53}$$

This shows that V contains the eigenvectors of the row-wise correlation matrix and U contains the eigenvectors of the column-wise correlation matrix. The matrix Σ contains the roots of the according eigenvalues and is thereby related to the variance. By ordering U, Σ and V by the entries of Σ in a descending order he first row of U and V contain the most important basis vectors [6].

2.3.3. Low-rank approximation

A useful property of the SVD is that it can be used to find an hierarchy of rank-r approximation for a given matrix X. An matrix $\tilde{X}$ that approximates X is obtained by

$$\tilde{X} = \underset{s.t. rank(\tilde{X})=r}{\operatorname{argmin}} \|X - \tilde{X}\|_F = \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{2.54}$$

Here $\tilde{U}$ and $\tilde{V}$ denote matrices obtained taking the first r columns of U and V. The matrix $\tilde{\Sigma}$ is a $r \times r$ sub-block of Σ. This is also known as the Eckard-Young theorem. The variance captured by $\tilde{X}$ can be calculated in the following way [6]

$$\text{cumvar}_r(\Sigma) = \frac{\sum_{i=1}^{r} \sigma_i}{\text{trace}(\Sigma)} \tag{2.55}$$

$$\text{var}(\Sigma) = \frac{\text{diag}(\Sigma)}{\text{trace}(\Sigma)}\,. \tag{2.56}$$

2.3.4. Example low-rank approximation

As an example suppose there is a matrix X

$$X = \begin{bmatrix} 3 & 1 & 5 & 5 \\ 4 & -4 & 5 & 0 \\ -4 & -2 & -4 & 3 \\ 5 & 1 & 5 & -4 \end{bmatrix}. \tag{2.57}$$

By computing the SVD the matrices U, Σ and V are obtained. Now Σ can be used to calculate the cumulative variance for an rank-r approximation and the variance captured by each basis vector of U and V.

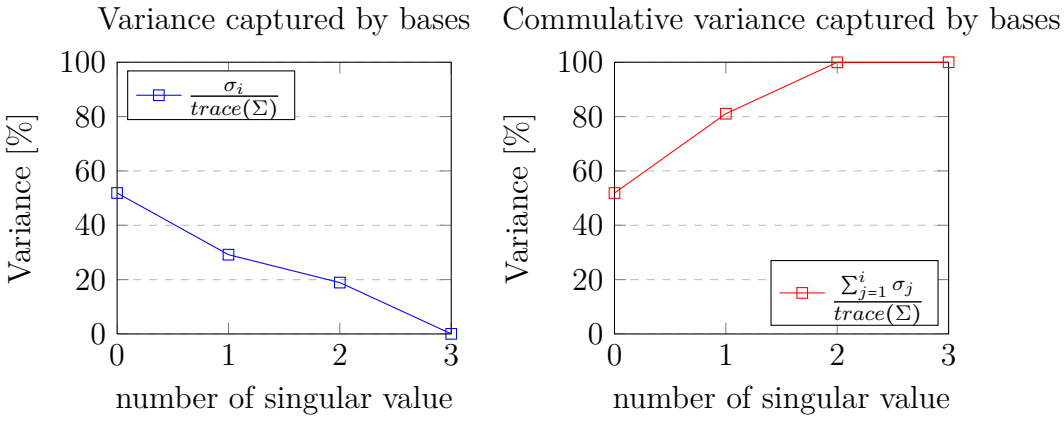


Figure 2.1.: Variance and cumulative variance captured by each column vectors of U and V

On figure 2.1 the commutative variance and the variance of each basis vector is plotted. Here X can be approximated using the first three leading basis vectors. This approximation captures already more than 99% of the variance. The resulting matrix $\tilde{X}_3$ looks as follows

$$\tilde{X}_3 = \begin{bmatrix} 4.73 & -0.76 & 5.62 \\ 1.31 & -1.37 & 1.62 \\ -5.10 & -0.60 & 2.34 \end{bmatrix}. \tag{2.58}$$

2.4. Fundamentals of Control Theory

There are mainly three different problems in control theory. To analyse a system it has to be identified first. A mathematical model has to be found that relates the input of that system to it's output. After a model has been found, it is helpful to simulate the system to find out which input results in which output. Simulations have to fit the system and offer the advantage over real experiments of reducing cost and risks attached to them. The last of the three major problems is the control problem. Now that the underlying model is known and the system can be simulated, a controller for that system can be designed. Here the controller is a system that feeds some input to the system that is to be controlled to generate some desired output [12]. For this paper the second problem is the most important one, since the goal of this paper is to compare some methods to make simulations of a system more efficient.

2.4.1. LTI Systems

A linear time invariant (LTI) system is a kind of system that fulfils the following conditions:

Homogenity: For a system to be a LTI system, the condition that the output of a system $y(t)$ relates to the input $u(t)$ and the initial condition x_0 by a linear operator $\mathfrak{f}$

$$y(t) = \mathfrak{f}(u(t), x_0) \tag{2.59}$$
$$cy(t) = \mathfrak{f}(cu(t), x_0). \tag{2.60}$$

This means that if the input u changes in magnitude by a constant factor of c the output y also changes by the factor of c.

Superposition: The second requirement is superposition

$$y_1(t) + y_2(t) = \mathfrak{f}(u_1(t) + u_2(t), x_0). \tag{2.61}$$

The sum of two outputs y_1 and y_2 has to equal the output of the system with the sum of the inputs u_1 and u_2 as input.

Time Invariance: A system that is time invariant has the property that if the input u is shifted in time by some constant τ the resulting output is also shifted in time by the same constant

$$y(t-\tau) = \mathfrak{f}(u(t-\tau), x_0)\,. \tag{2.62}$$

LTI systems are important because they can be used to approximate non LTI systems over some region. This is useful since LTI systems are well understood [13].

2.4.2. State Space Representation

A LTI system can be expressed as a system of ODEs that relates the input u to its output y. The vector x denotes the states of a system.

$$\dot{x} = Ax + Bu \quad x(t_0) = x_0 \tag{2.63}$$
$$y = Cx + Du \tag{2.64}$$

The matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{q\times n}$ and $D \in \mathbb{R}^{q\times m}$ are constant matrices. [14]

2.4.3. State Space of Discrete Time Systems

A system can represented as discrete system in the following form

$$x_{k+1} = A_d x_k + B_d x_k \tag{2.65}$$
$$y_{k+1} = C_d x_x + D_d u_k \tag{2.66}$$
$$x_k = x_{k\Delta t}\,. \tag{2.67}$$

The matrices A_d to D_d can be obtained from the continuous system in the following way [15]

$$A_d = e^{A\Delta t}, \quad B_d = \int_0^{\Delta t} e^{A\tau} B d\tau, C_d = C, \quad D_d = D\,. \tag{2.68}$$

2.4.4. Transfer Function

To understand the transfer function the impulse response has to be defined first. The impulse response of a system is given by choosing the delta function as input to the system

$$\delta(t) = \begin{cases} \infty, & t = 0 \\ 0 & else \end{cases} \quad g(t) = f(\delta(t))\,. \tag{2.69}$$

It has the property that the integral of this function is 1

$$\int_{t=-\infty}^{\infty} \delta(t)dt = 1\,. \tag{2.70}$$

If the $g(t)$ is convoluted with a function $v(t)$ the response of the system for $v(t)$ as input is obtained [16]

$$v(t) * g(t) = \int_{\tau=0}^{\infty} v(\tau)g(t-\tau)d\tau = f(v(t))\,. \tag{2.71}$$

By using the convolution theorem [17]

$$\mathfrak{L}\{g(t)\}\mathfrak{L}\{v(t)\} = \mathfrak{L}\{g * v\}(t)\,. \tag{2.72}$$

the transfer function is defined as

$$\mathfrak{L}\{f(\delta(t))\}\,. \tag{2.73}$$

This definition is useful since the response of a system for a given input $u(t)$ can be determined by the result of the multiplication $G(s)U(s)$ [16]. Alternatively if a system is given in state space representation the transfer function can be expressed as

$$G(s) = C(sI - A)^{-1}B + D\,. \tag{2.74}$$

[14]

2.4.5. Controllability and Observability

For LTI systems in state space representation it can be determined which states are to what degree controllable and observable. This can be computed by the controllability and observability gramians

$$W_c = \lim_{t\to\infty} \int_0^t e^{A\tau}BB^*e^{A^*\tau}d\tau \tag{2.75}$$

$$W_o = \lim_{t\to\infty} \int_0^t e^{A^*\tau}C^*Ce^{A\tau}d\tau\,. \tag{2.76}$$

The degree of controllability for a state x can be determined by x^*W_cx. If the result of this calculation is large, the system is controllable in the x direction. By swapping W_c with W_o the degree of observability in state x can be computed: x^*W_ox. Again if the resulting value is large, the system can be observed well in state x [18].

3. Model Order Reduction

3.1. Introduction

Model Order Reduction (MOR) is a technique to reduce the computational effort of simulating a system using mathematical models. This is done by using two different approaches. The first one is using a numerical approach and the second approach is based on system theory. The first method discussed is called Proper Orthogonal Decomposition. After that Balanced and Modal Truncation will be discussed. The last method discussed will be Hankel-norm approximation [19].

3.2. Proper Orthogonal Decomposition

The proper orthogonal decomposition (POD) is a method for model order reduction. The reduction in computational effort is done by approximating a solution to a PDE using an orthogonal expansion. The basis functions are obtained by decomposing a set of solutions for the PDE using the SVD.

3.2.1. Orthogonal Expansion

A function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ can be represented by the series

$$f(x,t) = \sum_{i=1}^{\infty} a_i(t)\phi_i(x)\,. \tag{3.1}$$

Here all $\phi(x)$ are orthogonal basis functions

$$\langle \phi_i, \phi_j \rangle = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{else} \end{cases}\,. \tag{3.2}$$

By using a finite sum instead of the entire series f can be approximated

$$\tilde{f}(x,t) = \sum_{i=1}^{n} a_i(t)\phi_i(x)\,. \tag{3.3}$$

Since all ϕ are known, (3.3) has to be solved for the set $\{a_0, ..., a_n\}$. In case f is known, the solutions contained in this set can be calculated as

$$a_i(t) = \frac{\langle \phi_i, f \rangle}{\langle \phi_i, \phi_i \rangle} . \tag{3.4}$$

[20]

3.2.2. Proper Orthogonal Decomposition for PDEs

Since a PDEs are equations in terms of partial derivatives, the notation $\mathfrak{P}(\partial x)u$ is introduced, which denotes a differential operator in terms of the spatial variables x for a function $u(x, t, p.$ The vector p contains some parameters. This is done to provide a more abstract way to denote PDEs

$$\frac{\partial u}{\partial t} = \mathfrak{P}(\partial x)u . \tag{3.5}$$

[21]
Solving for u is often difficult to impossible. A method that is often used to solve PDEs is called separation of variables. This separation of variables assumes, that the underlying solution $u(x, t)$ can be expressed by 3.1, to solve for $a_k, 0 \leq k \leq n$. Since it is not practical to compute an infinite series, u only gets approximated by using 3.3 instead

$$\sum_{i=1}^{n} \frac{\partial a_i}{\partial t}\phi_i = \sum_{i=1}^{n} \mathfrak{P}(\partial x)\phi_i a_i . \tag{3.6}$$

By discretizing the spatial dimension got along x 3.6 can be expressed using matrix notation

$$\Phi = \begin{bmatrix} \phi_0 & ... & \phi_n \end{bmatrix} \tag{3.7}$$

$$\Phi \frac{d}{dt} a = \mathfrak{P}(\partial x)\Phi a . \tag{3.8}$$

Since the solution u is unknown 3.4 cannot be computed. However, the fact, that all ϕ are orthogonal to each other, can be used to solve for all a_k. It can be done by computing the inner product of 3.6 with all basis functions

$$\left\langle \sum_{i=1}^{n} \frac{\partial a_i}{\partial t}\phi_i, \phi_k \right\rangle = \left\langle \sum_{i=1}^{n} \mathfrak{P}(\partial x)\phi_i a_i, \phi_k \right\rangle \quad , \quad 0 \leq k \leq n . \tag{3.9}$$

This resembles the Galerking projection. In matrix notation it can be expressed

$$\Phi^T \Phi \frac{d}{dt} a = \Phi^T \mathfrak{P}(\partial x)\Phi a . \tag{3.10}$$

By considering 3.2 the equation 3.9 can be expressed as a system of ODEs which can be solved

$$\frac{d}{dt}a = \Phi^T \mathfrak{P}(\partial x)\Phi a\,. \tag{3.11}$$

After the vector of coefficients a for each time step has been computed, the solution can be assembled [22]

$$u(x,t) \approx \Phi(x)a(t)\,. \tag{3.12}$$

3.2.3. Selection of Basis Vectors

As discussed in the previous section, a set of basis vectors can be used to generate approximate solutions to PDEs. However it was not discussed how those basis functions are chosen. For POD to work, a so-called snapshot matrix X has to be available. This snapshot matrix stores a set of solutions where x_k is the solution for a PDE at time step $k\Delta t$

$$X = \begin{bmatrix} x_1 & \dots & x_m \end{bmatrix}. \tag{3.13}$$

The solutions can be obtained by conducting an experiment on a physical system that is described by the PDE or by simulating the evolution of that PDE. In this paper the solution is obtained using FEM 2.2. SVD is used to decompose the snapshot matrix X. Since the columns of the snapshot matrix contain spatial information at a given point in time and the basis vectors are supposed to encode the spatial information of a solution u the r most dominant left singular values have to be extracted

$$\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{3.14}$$

$$\Phi = \begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix}. \tag{3.15}$$

Here the FEM solution is obtained by discretizing the PDE into a large number(n) of spatial nodes. This results in a high dimensional system of ODEs. Since Φ contains only r column ($r << n$) vectors a reduction in order can be achieved. The number of modes r is determined by the variance that has to be persevered. Figure 5.2 shows the variance each mode stores. Here the first two modes alread store almost 97% of the variance.

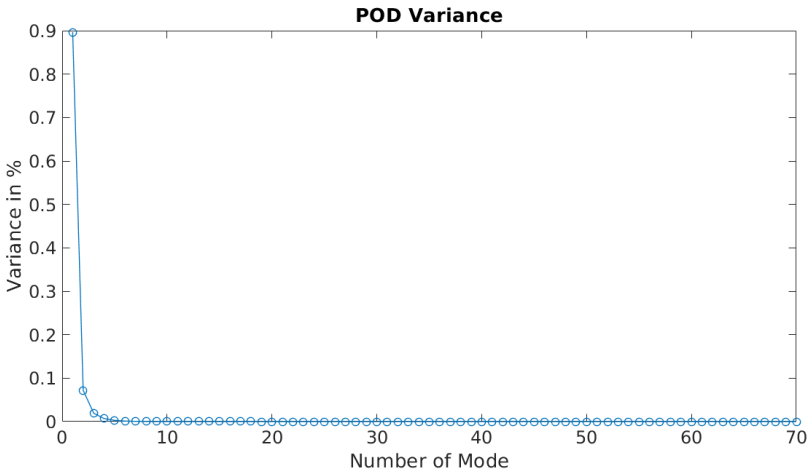


Figure 3.1.: Variance of Modes

These two modes are displayed on 3.2.

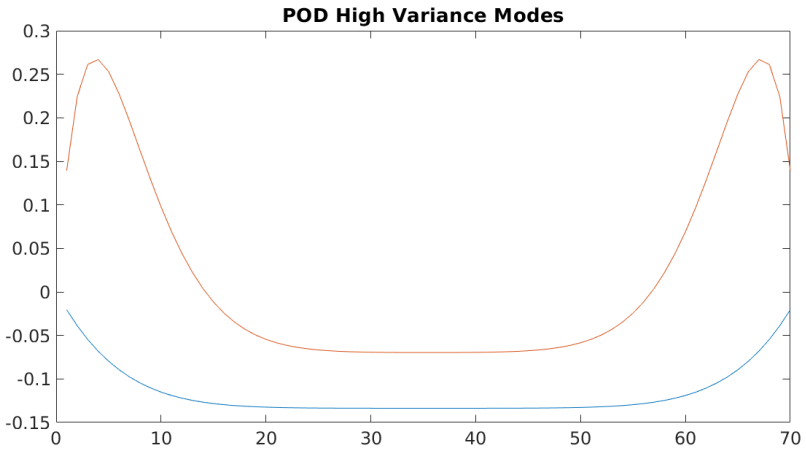


Figure 3.2.: Leading High Variance Modes

Setting the variance too high inlcudes also low variance modes that will introduce some errors. Some low variance modes are displayed on 3.3.

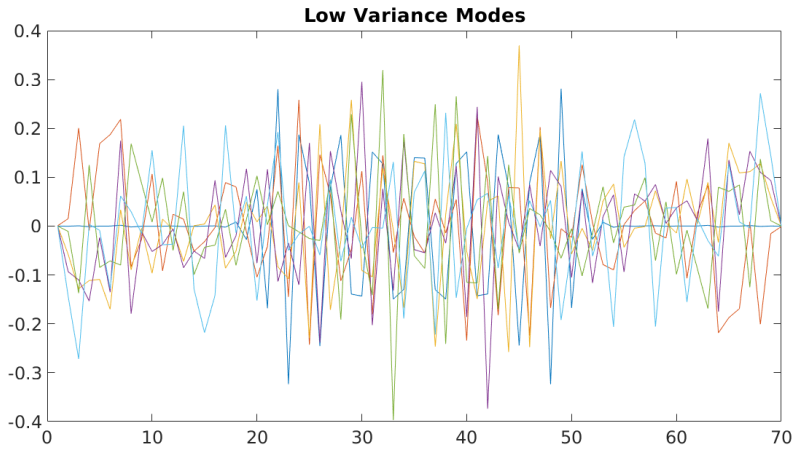


Figure 3.3.: Modes 30 to 35

Figure 5.2 shows that modes 30 to 35 almost have zero contribution to the snapshot matrix. Therefore including them will introduce noise to the approximation and will increase computations. However the benefits of this reduced order model are only relevant after the PDE has been solved once using a high dimensional system of ODEs [22].

3.2.4. POD for Heat Equation

In order to apply pod to heat equation a snapshot matrix X has to be generated. This is done by the FEM solver described in section 2.2. After that X gets decomposed using the SVD. The modes contained in Φ is obtained by truncating the left singular vectors of X according to (3.15). Substituting $\mathfrak{P}$ in (3.12) with heat equation 2.6 results in the following

$$\Phi \frac{\partial a}{\partial t} = \alpha \frac{\partial^2 \Phi}{\partial x^2} a + h \tag{3.16}$$

$$\frac{\partial a}{\partial t} = \alpha \Phi^T \frac{\partial^2 \Phi}{\partial x^2} a + \Phi^T h\,. \tag{3.17}$$

This system of ODEs can now be solved using a Runge-Kutta scheme. Note that Φ contains numeric values. Therefore derivatives are unstable, especially at the first and last entries of the column vectors of Φ. To reduce this problem the method used to compute the second derivative of Φ is the so-called spectral derivative. The discrete spectral derivative works by computing the FFT of a vector. That vector is multiplied by $(ik)^d$, d is the order of the derivative and k are the discrete wave numbers. After that step the IFFT is applied to obtain

the derivative of the original vector [23]

$$f \in \mathbb{C}^n, \quad k = \begin{bmatrix} \frac{n}{2} & \cdots & \frac{n}{2} \end{bmatrix}^T \tag{3.18}$$

$$\frac{df}{dx} = \mathfrak{F}^{-1}\left\{ i\frac{2\pi k}{n}\mathfrak{F}\{f\} \right\} \tag{3.19}$$

$$\frac{d^2 f}{dx^2} = \mathfrak{F}^{-1}\left\{ -\frac{2\pi k}{n}\mathfrak{F}\{f\} \right\}. \tag{3.20}$$

Figure 3.4 shows a solution to heat equation using FEM and an approximation using POD with $u(x,t) = 0, x_0 = 10000^{70\times 1}, n = 70,$ variance = 90%. The x-axis displays time in seconds and the y-axis the length of the object. It is clearly visible that there are some differences, especially near the edges. However a more detailed analysis of the results are covered in chapter 5.

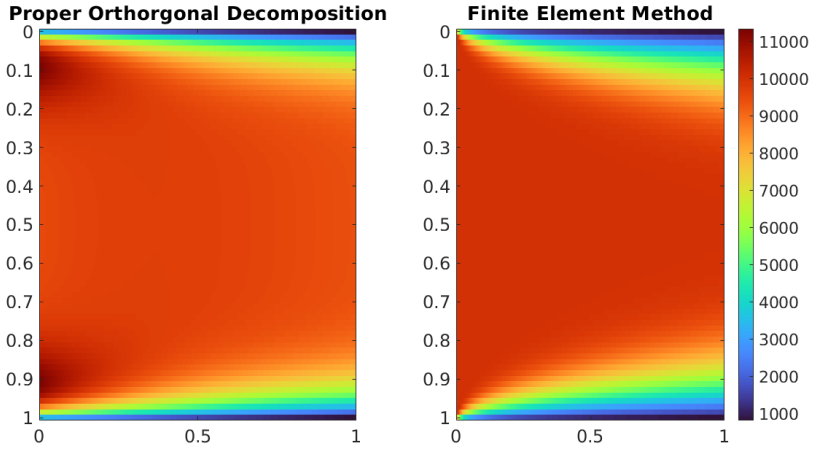


Figure 3.4.: FEM solution and POD approximation for heat equation

3.3. Balanced Truncation

Balanced truncation is a method for model order reduction. The goal of balanced truncation is to approximate a system using only the most relevant modes of the system. The difference to POD is that the modes are not selected by the variance they capture but by the controllability and observability of the modes. This is done by finding a coordinate transform.

3.3.1. Balanceing Coordinate Transform

To find a reduced order model using balanced truncation a orthonormal coordinate transform is applied: $x = Tz$. This yields a new system

$$\dot{z} = \hat{A}z + \hat{B}u \tag{3.21}$$

$$y = \hat{C}z + Du \tag{3.22}$$

$$\hat{A} = T^{-1}AT \quad \hat{B} = T^{-1}B \quad \hat{C} = CT\,. \tag{3.23}$$

The gramians of this ROM can be obtained by applying 2.76 and 2.75 to 3.23. This yields $\hat{W}_c = T^{-1}W_cT^{-*}$ and $\hat{W}_o = T^*W_oT$ with $T^{-*} := (T^{-1})^* := (T^*)^{-1}$. A requirement T has to satisfy is that it has to make the observability and controllability gramians of the ROM equal and diagonal

$$\hat{W}_c = \hat{W}_o = \Delta \tag{3.24}$$

$$\hat{W}_c\hat{W}_o = \Delta^2 \tag{3.25}$$

$$T^{-1}W_cW_oT = \Delta^2 \tag{3.26}$$

$$W_cW_oT = T\Delta^2\,. \tag{3.27}$$

Since Δ is a diagonal matrix, 3.27 is equal to the eigendecomposition of W_cW_o. Therefore T contains the eigenvectors of W_cW_o. The values contained in Δ are known as hankel singular values. However T needs to be rescaled to make $\hat{W}_c$ and $\hat{W}_o$ equal. Here T_u denotes the unscaled eigenvectors of this eigendecomposition that yields gramians that are not equal to each other

$$T_u^{-1}W_cT_u^{-*} = \Delta_c \tag{3.28}$$

$$T_u^*W_cT_u = \Delta_o\,. \tag{3.29}$$

Scaling T_u by some diagonal matrix Δ_s results in $\Delta_c = \Delta_o$

$$\Delta_s = \Delta_c^{\frac{1}{4}}\Delta_o^{-\frac{1}{4}} \tag{3.30}$$

$$T = T_u\Delta_s\,. \tag{3.31}$$

Another important property of this transform is that the new coordinates are hierarchically ordered by observability and controllability. It can be shown by deriving some unit vector ζ that maximizes the controllability and observability

$$\zeta = arg\max\, \zeta^*W_cW_o\zeta \quad s.t.\|\zeta\|_2^2 = 1 \tag{3.32}$$

$$\frac{d}{d\zeta}\zeta^*W_cW_o\zeta - 2\lambda\zeta = 0\,. \tag{3.33}$$

As shown here [24] the remaining derivative can be solved in the following way

$$\frac{d}{dx}x^*Ax = 2Ax\,. \tag{3.34}$$

This holds if A is symmetric. Here both W_c and W_o share the same set of eigenvectors A.15 and 3.29, therefore they commute [25]. This means that the product of W_c and W_o is also symmetric [26]. Applying 3.34 to 3.33 yields

$$W_cW_o\zeta = \lambda\zeta\,. \tag{3.35}$$

Since λ is the Lagrange multiplier, it is a scalar. It is clear that ζ is a eigenvector. Since the eigenvalues in Δ contain information about how much each eigenvector gets scaled by multiplying it with W_cW_o the eigenvectors can be ordered by controllability and observability.

3.3.2. Mode Truncation

Since the goal of balanced truncation is to find a ROM of rank r that approximates the original system of rank n with $r << n$ it is necessary to truncate the balanced system. This yields the following system:

$$\frac{d\tilde{x}}{dt} = \tilde{A}\tilde{X} + \tilde{B}u \tag{3.36}$$

$$y = \tilde{C}\tilde{x} + \tilde{D}u\,. \tag{3.37}$$

The new state vector $\tilde{x}$ is defined as

$$\tilde{x} = \begin{bmatrix} z_1 \\ \vdots \\ z_r \end{bmatrix} \quad \tilde{z} = \begin{bmatrix} z_{r+1} \\ \vdots \\ z_n \end{bmatrix} \quad z = \begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix} \tag{3.38}$$

$$T = \begin{bmatrix} \Psi & T_t \end{bmatrix} \quad T^{-1} = S = \begin{bmatrix} \Phi^* \\ S_t \end{bmatrix}. \tag{3.39}$$

By substituting 3.38 and 3.39 into the system in 3.21 and 3.22 the system becomes

$$\frac{d}{dt}\begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \Phi^*A\Psi & \Phi^*AT_t \\ S_tA\Psi & S_tAT_t \end{bmatrix}\begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix} + \begin{bmatrix} \Phi^*B \\ S_tB \end{bmatrix}u \tag{3.40}$$

$$y = \begin{bmatrix} C\Psi & CT_t \end{bmatrix}\begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix} + Du\,. \tag{3.41}$$

However the only relevant part of this system is

$$\frac{d\tilde{x}}{dt} = \Phi^* A \Psi \tilde{x} + \Phi^* B u \tag{3.42}$$

$$y = C \Psi \tilde{x} + D u \tag{3.43}$$

since this is the only part necessary to calculate $\tilde{x}$ [27].

3.3.3. Computing Balanced Truncation

Since the gramians for controllability and observability are too expensive to compute for large systems the so called empirical gramians are used as an approximation. The empirical gramians are calculated by using the discrete-time system matrices from 2.65 and 2.66

$$\mathscr{C}_d = \begin{bmatrix} B_d & A_d B_d & \cdots & d A^{m_o - 1} B_d \end{bmatrix} \tag{3.44}$$

$$\mathscr{O}_d = \begin{bmatrix} C_d \\ C_d A_d \\ C_d A_d^{m_o - 1} \end{bmatrix} \tag{3.45}$$

$$W_c^e = \mathscr{C}_d^* \mathscr{C}_d \tag{3.46}$$

$$W_o^e = \mathscr{O}_d^* \mathscr{O}_d \tag{3.47}$$

$$m_o, m_c << rank(A)\,. \tag{3.48}$$

Now these empirical gramians can be used for obtaining the balancing coordinate transform [27].

3.3.4. State Space Representation from Heat Equation

To apply balanced truncation to the heat equation 2.6 a state space representation of that system has to be found first. Note that the system of ODEs resulting from FEM 2.39 resembles a state space representation

$$x := c \quad x_0 := c_0 \tag{3.49}$$

$$A := M^{-1} K \quad B := I\,. \tag{3.50}$$

Since the system is realized as simulation all states can be accurately measured $C := I$ and there is no feed through $D := 0_{qm}$. Now the already described steps for balanced truncation can be applied to the system.
Figure 3.5 shows a solution to heat equation using FEM and an approximation using BT with $u(x,t) = 0, x_0 = 10000^{70\times 1}, n = 70, n_{approx} = 10$.

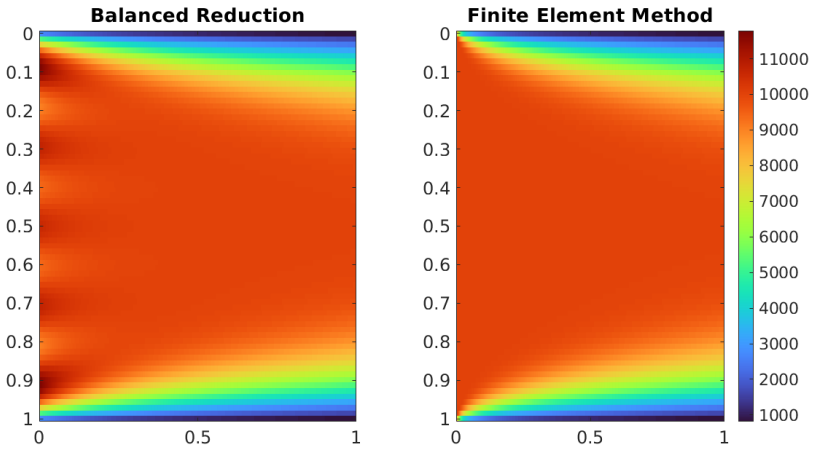


Figure 3.5.: FEM solution and BT approximation for heat equation

3.4. Modal truncation

3.4.1. Diagonal Canonical Form

A LTI system (A, B, C, D) with transfer function $G(s)$ can be written in the following way

$$I = \{1, 2, \cdots, n\} \tag{3.51}$$

$$G(s) = D + \sum_{i \epsilon I} \frac{\Phi_i}{s - \lambda_i} \,. \tag{3.52}$$

The set $\Lambda = \{\lambda_1, \cdots, \lambda_n\}$ denotes the eigenvalues of the matrix A in state space. This resembles the partial fraction decomposition of $G(s)$ [28]. The residue Φ can be computed using the eigendecomposition of A and using the eigenvectors for a coordinate transform

$$AX = X\Delta \quad X\zeta = x \,. \tag{3.53}$$

The system matrices have to be transformed accordingly

$$\hat{A} = X^{-1}AX = \Delta \tag{3.54}$$

$$\hat{B} = X^{-1}B \quad \hat{C} = CX \tag{3.55}$$

$$\hat{D} = D \,. \tag{3.56}$$

Those transformed matrices are used to calculate the transfer function 2.74

$$G(s) = CX(sI - \Delta)^{-1}X^{-1}B + D \tag{3.57}$$

$$= \begin{bmatrix} Cx_1 & \cdots & Cx_n \end{bmatrix} diag\left\{\frac{1}{s-\lambda_1}, \cdots, \frac{1}{s-\lambda_n}\right\} \begin{bmatrix} x_1^{-1}B \\ \vdots \\ x_n^{-1}B \end{bmatrix} + D \tag{3.58}$$

$$= \sum_{i \in I} \frac{c_i b_i^T}{s-\lambda_i} + D \Rightarrow \Phi_i = c_i b_i^T \,. \tag{3.59}$$

[29]

3.4.2. Optimal Modal Truncation

The goal of modal truncation is to find a subset of the indices I such that only r elements are contained in this subset

$$I_r \subseteq I, \quad |I_r| = r \,. \tag{3.60}$$

Using this subset as indices in 3.52 a truncation is obtained, yielding a system defined by a new transfer function $\hat{G}(s)$. The set I_r has to be chosen such that the error $\|G(s) - \hat{G(s)}\|_{H_n}$ becomes minimal. Here H_n denotes the H_2 norm. As shown in [28] other norms are also usable but for simplicity only the stated norm is used. Furthermore in case some $\lambda_i \in \Lambda, \mathrm{Im}(\lambda_i) > 0$ is a complex number $i \in I_r$, the index of the according complex conjugate eigenvalue has to selected too, if $\lambda_j = \bar{\lambda}_i \in \Lambda \wedge i \in I_r \Rightarrow j \in I_r$. This yields the following optimization problem

$$G_\alpha(s) = D_\alpha + \sum_{i \in I} \alpha_i \frac{\Phi_i}{s-\lambda_i} \tag{3.61}$$

$$\min_\alpha \|G(s) - G_\alpha(s)\| s.t. \quad \alpha^T \alpha = r \,. \tag{3.62}$$

With the following constrains

$$J = \{(\lambda_i, \lambda_j) \in I | \lambda_i \in \mathbb{C}, \mathrm{Im}(\lambda_i) > 0, \lambda_i = \bar{\lambda}_j\} \tag{3.63}$$

$$a_{.,j} = -a_{.,l} = 1, \quad (i,j) \in J \tag{3.64}$$

$$A\alpha = 0 \,. \tag{3.65}$$

Where $\alpha \in \{0,1\}^n$ is some binary vector with $\alpha^T\alpha = r$ that represents the selection of indices. By defining the error of that system as

$$\epsilon_\alpha(s) = G(s) - G_\alpha \tag{3.66}$$

$$\|\epsilon_\alpha(s)\|_2^2 = \sum_{i,k\in I}(1-\alpha_i)\frac{tr(\phi_i\phi_k^T)}{-\lambda_i-\lambda_k}(1-\alpha_k) \tag{3.67}$$

$$\|\epsilon_\alpha(s)\|_2^2 = (1-\alpha)Q(1-\alpha) \tag{3.68}$$

$$q_{ij} = \frac{tr(\phi_i\phi_k^T)}{-\lambda_i-\lambda_k^*}\,. \tag{3.69}$$

This leads to a new optimization problem [28]

$$\min_\alpha \quad (1-\alpha)Q(1-\alpha) \tag{3.70}$$

$$s.t. \quad \alpha^T\alpha = r \tag{3.71}$$

$$A\alpha = 0\,. \tag{3.72}$$

3.4.3. Applying Modal Truncation to Heat Equation

As described in 3.3.4 the system matrices (A, B, C, D) can be obtained for the given heat equation. Now by applying 2.74 yields the corresponding transfer function $G(s)$. By calculating the partial fraction decomposition as described in 3.52 the DCNF can be obtained. Since B and C are given by some identity matrix and D is a zero matrix the upper error bound of the error in H_2 norm is given as [28]

$$\|\epsilon(s)\|_{H_2} \leq \sum_{i\in I_r^c}\frac{1}{\sqrt{-2Re(\lambda_i)}}\,. \tag{3.73}$$

Since for stable systems $Re(\lambda) < 0 \forall \lambda \in \Lambda$ this upper bound can be expressed as follows with $\Lambda_\epsilon = \{Re(\lambda_i)|\lambda_i \in \Lambda i \in I_r^c\}$

$$\|\epsilon(s)\|_{H_2} \leq \sum_{i\in I_r^c}\frac{1}{\sqrt{2|Re(\lambda_i)|}} \leq \frac{|I_r^c|}{\sqrt{2|\max(\Lambda_\epsilon)|}}\,. \tag{3.74}$$

To minimize this upper bound $|\max(\Lambda_\epsilon)|$ has to be maximized. This is achieved by selecting the entries of Λ_ϵ such that it contains $n-r$ largest real parts of the eigenvalues in Λ.
Figure 3.6 shows a solution to heat equation using FEM and an approximation using MT with $u(x,t) = 0, x_0 = 10000^{70\times1}, n = 70, n_{approx} = 10$.

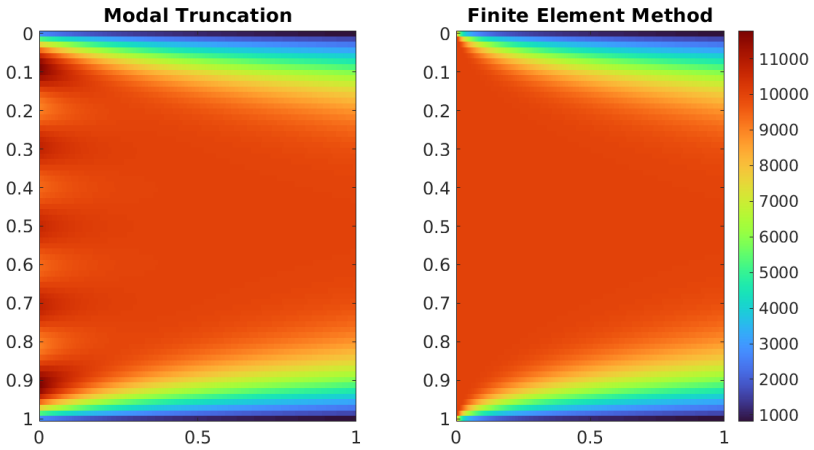


Figure 3.6.: FEM solution and MT approximation for heat equation

3.5. Hankel Norm Approximation

The goal of the Hankel Norm Approximation (HNA) is to find a reduced order system that approximates a system G such that the error in the so called Hankel norm becomes minimal. The Hankel norm is defined to be the largest hankel singular value

$$\|G(s)\|_H = \sigma_{max} = \sqrt{\lambda_{max}(W_c W_o)} \tag{3.75}$$

[30]. Therefore the problem can be stated as follows

$$G_r = arg\min \|G - G_r\|_H \tag{3.76}$$

3.5.1. System Spaces

For the following section it is necessary to define the four following system spaces: L_∞, H_∞, H_∞^- and $H_\infty^-(r)$.

L_∞ $G \in L_\infty$ iff $\sup_\omega \|G(i\omega)\| < \infty$.

H_∞ $G \in H_\infty$ iff $\forall \lambda \in \mathbb{C}_-$. Here λ are the eigenvalues of A.

H_∞^- $G \in H_\infty^-$ iff $G(-s) \in H_\infty$.

$H_\infty^-(r)$ $G \in H_\infty^-(r)$ iff $G \in L_\infty$ and $\lambda = \lambda_+ \cap \lambda_- = \{\lambda_1, ..., \lambda_n\}$ with $\lambda_\circ = \{\lambda_i \in \lambda | \lambda_i \in \mathbb{C}\_\circ\}$ and $|\lambda_-| \leq r$.

3.5.2. Optimal Solution

A lower bound for $\min \|G - G_r\|_H$ is established by lemma 7.1 in [31]

$$\min \|G - G_r\|_H \geq \sigma_{r+1} \tag{3.77}$$

.
Here σ_{r+1} is the $r+1$th largest hankel singular value of G. Hence a system G_r is optimal if $\|G - G_r\|_H = \sigma_{r+1}$.
It is important to note that the Hankel norm can is related to the L_∞ norm through the Nehari Theorem

$$G \in H_\infty, \quad F \in H_\infty^-, \quad G - F \in Ł_\infty \tag{3.78}$$

$$\|G\|_H = \min_{F \in H_\infty^-} \|G - F\|_\infty \tag{3.79}$$

.
Also the Adamjan-Arov-Krein theorem has to be stated to find a optimal solution to the stated minimization problem

$$G \in H_\infty, \quad Q \in H_\infty^-(r), \quad G - Q \in L_\infty \tag{3.80}$$

$$\min_{Q \in H_\infty^-} \|G - Q\|_\infty = \sigma_{r+1} \tag{3.81}$$

.
Suppose there is an optimal system $Q^* = G_r + F$ with $Q^* \in H_\infty^-(r), G_r \in H_\infty, F \in H_\infty^-$. It has the following error bound

$$\|G - G_r\|_\infty = \|G - Q^* + F\|_\infty \leq \sigma_{r+1} + \|F\|_\infty. \tag{3.82}$$

If $\|F\|_\infty$ is small enough, the stable part of Q^* can be used as a reduced order model. It is also an optimal solution to 3.76

$$\|G - G_r\|_H = \min_{F \in H_\infty^-} \|G - G_r - F\|_\infty = \min_{Q \in H_\infty^-} \|G - Q\|_\infty = \sigma_{r+1} \tag{3.83}$$

.
[32]

3.5.3. Constructing Q^*

The first step to construct an optimal system Q^* is to construct an balanced realization of the system $G = (A, B, C, D) \in H_\infty$that is to be reduced, as described in section 3.3.1. Hence the gramians W_c and W_o are equal and diagonal.

$$W_c = \begin{bmatrix} P_1 & 0 \\ 0 & \sigma_{r+1} I_l \end{bmatrix}, \quad W_o = \begin{bmatrix} Q_1 & 0 \\ 0 & \sigma_{r+1} I_l \end{bmatrix} \tag{3.84}$$

Now G is partitioned in the following way

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \tag{3.85}$$

.
Also a unitary matrix U has to be defined such tat $B_2 = -C_2^T U$ and $U^T U = I$. Further more a matrix $E_1 = P_1 Q_1 - \sigma_{r+1}^2 I$ is introduced.
Then a system $Q^* = (\hat{A}, \hat{B}, \hat{C}, \hat{D})$ can be defined

$$\hat{A} = E_1^{-1}(\sigma_{r+1}^2 A_{11}^T + Q_1 A_{11} P_1 - \sigma_{r+1} C_1^T U B_1^T) \tag{3.86}$$

$$\hat{B} = E_1^{-1}(Q_1 B_1 + \sigma_{r+1} C_1^T U) \tag{3.87}$$

$$\hat{C} = C_1 P_1 + \sigma_{r+1} U B_1^T \tag{3.88}$$

$$\hat{D} = D - \sigma_{r+1} U \tag{3.89}$$

. [32]

3.5.4. Decomposition of Q^*

The final step is to decompose Q^* into two systems $G_r \in H_\infty, F \in H_\infty^-$. To achieve this Q^* can be expressed in the diagonal canonical form described in section 3.52.

$$Q^* = D + \sum_{i=1}^{r} \frac{\phi_i}{s - \lambda_i} \tag{3.90}$$

.
This can no be decomposed into three systems $K \in H_\infty, V \in H_\infty^-$ and $\tilde{D}$ with

$$Re(\lambda_i) \in \mathbb{C}_{circ} \forall i \in I_\circ \tag{3.91}$$

$$K = \sum_{i \in I_-} \frac{\phi_i}{s - \lambda_i} \tag{3.92}$$

$$V = \sum_{i \in I_+} \frac{\phi_i}{s - \lambda_i} \tag{3.93}$$

$$\tilde{D} = \hat{D} \tag{3.94}$$

.
Since $\tilde{D}$ is some constant it does not have any poles. Therefore $K + D \in H_\infty$ and $V + D \in H_\infty^-$ which leads to two different decompositions $Q^* = (K + \tilde{D}) + V$ and $Q^* = K + (V + \tilde{D})$. From 3.82 it is clear that $\|F\|_\infty$ has to be as small as possible to minimize the error bound. It can be shown that $Q^* = (K + \tilde{D}) + V$ is the optimal decomposition. Suppose $Q^* = K + (V + \tilde{D})$

was the optimal decomposition, then the according lower bound has to be smaller

$$\|G - K\|_\infty \leq \|G - (K + \tilde{D})\|_\infty \tag{3.95}$$
$$\sigma_{r+1}\|V + \tilde{D}\|_\infty \leq \sigma_{r+1}\|V\|_\infty \tag{3.96}$$
$$\|V + \tilde{D}\|_\infty \leq \|V\|_\infty + \|\tilde{D}\|_\infty \geq \|V\|_\infty \tag{3.97}$$
$$\Rightarrow \|G - K\|_\infty \geq \|G - (K + \tilde{D})\|_\infty \tag{3.98}$$

.
Therefore $G_r = K + D$ and $F = V$.

3.5.5. Applying HNA to Heat Equation

3.6. Modal Truncation is Equal to Balanced Truncation

The results for BT and MT are the same since $B = C$ and A is symmetric. Remember that $A^{n\times n} = M^{-1}K$ where both K and M are symmetric, hence A is symmetric [26]. The eigenvectors of a symmetric matrix are mutually orthogonal [33]. Therefore the eigendecomposition of A is

$$AX = X\Delta \tag{3.99}$$

. The eigenvalues of a symmetric matrix are real . It is assumed that $\Delta = diag(\lambda_1, ..., \lambda_n)$ with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$. where $XX^T = X^TX = I$. By applying $x = X\zeta$ to the system, the system becomes

$$\dot{\zeta} = \Delta\zeta + X^Tu(t) \tag{3.100}$$
$$\zeta = X\zeta + Du(t) \tag{3.101}$$

The grammians of that systems are

$$W_c = \lim_{t\to\infty} \int_0^t e^{\Delta\tau} I e^{\Delta\tau} d\tau \tag{3.102}$$
$$= \lim_{t\to\infty} \int_0^t e^{2\Delta\tau} d\tau \tag{3.103}$$
$$= \lim_{t\to\infty} (e^{\Delta t} - I)\Delta^{-1} \tag{3.104}$$
$$W_o = \lim_{t\to\infty} \int_0^t e^{\Delta\tau} I e^{\Delta\tau} d\tau \tag{3.105}$$
$$= \lim_{t\to\infty} \int_0^t e^{2\Delta\tau} d\tau \tag{3.106}$$
$$= \lim_{t\to\infty} (e^{\Delta t} - I)\Delta^{-1} \tag{3.107}$$

Therefore $W_c = W_o$, where W_c and W_o are diagonal for stable systems. Since the transfer of heat is a stable process, it is assumed that the according system is stable.

$$W_c = W_o = \lim_{t\to\infty}(e^{\Delta t} - I)\Delta^{-1} \tag{3.108}$$

$$= -\Delta^{-1} \tag{3.109}$$

[34]

$$W_C = W_o = \begin{bmatrix} \frac{-1}{\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{-1}{\lambda_2} & 0 & \vdots \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{-1}{\lambda_n} \end{bmatrix} \tag{3.110}$$

This shows that the eigenvectors of A satisfy the conditions for balancing transformation T in section 3.3. Since the eigenvalues are all negative and real the order $\delta_{11} \geq \delta_{22} \geq ...\delta_{nn}$ applies. This enables the mode truncation described in section 3.3.

$$\tilde{x} = \begin{bmatrix} \zeta_1 \\ \vdots \\ \zeta_r \end{bmatrix} \quad \tilde{z} = \begin{bmatrix} \zeta_{r+1} \\ \vdots \\ \zeta_n \end{bmatrix} \quad z = \begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix} \tag{3.111}$$

$$X = \begin{bmatrix} X_r & X_{n-r} \end{bmatrix} \quad X^{-1} = \begin{bmatrix} X_r^* \\ X_{n-r} \end{bmatrix} \tag{3.112}$$

$$\frac{d\tilde{x}}{dt} = \Delta_r \tilde{x} + X_r^{-1} u \tag{3.113}$$

$$y = X_r \tilde{x} + Du \tag{3.114}$$

The transfer function of that system is

$$G(s) = X_r(sI - \Delta_r^{-1})X_r^T + D = X_r diag(\frac{1}{s-\lambda_1}, ..., \frac{1}{s-\lambda_r})X_r^T + D \tag{3.115}$$

$$= \sum_{i=1}^{r} \frac{1}{s-\lambda_i} + D \tag{3.116}$$

This resembles the DCNF of that system where the terms corresponding to the $n-r$ smallest eigenvalues are truncated. As shown in section 3.4.3 this is the optimal modal truncation. Therefore modal truncation and balanced truncation yields the same system.

4. Implementation

The implementation of the discussed methods for solving the heat equation and for model order reduction was done in Matlab. Matlab was chosen as the programming language

because it natively features matrix multiplication which finds heavy use in the previously mentioned methods. The second reason for this selection is that there exist ToolBoxes that already implement certain model order reduction methods such as MORLAB [35] or MOR toolbox [36]. The following figure 4.1 shows the class diagram of the implementation.

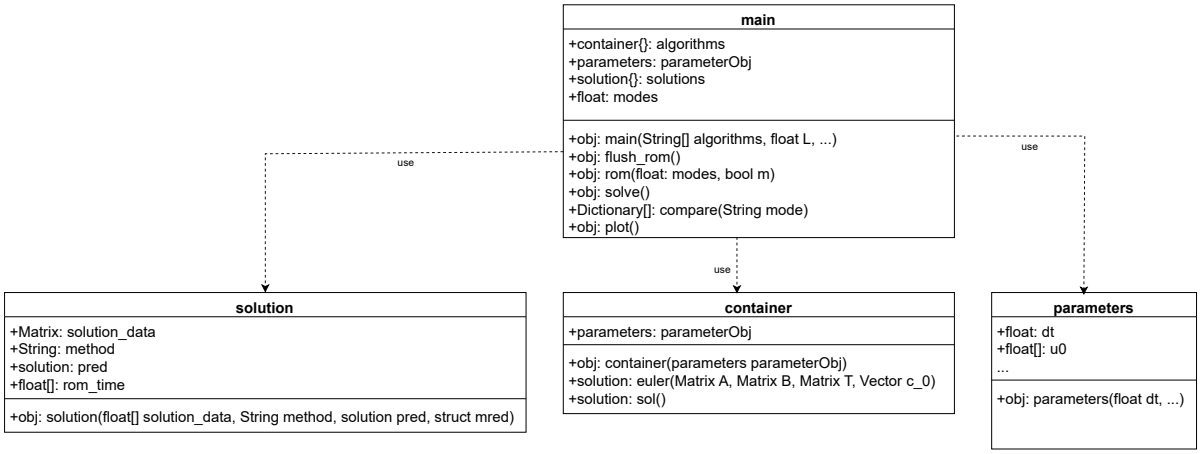


Figure 4.1.: Class diagram main

4.1. Class main

The class main is responsible for generating the finite element solution, model order reduction steps, plotting and comparing the results. The process can be seen in the following flow chart:

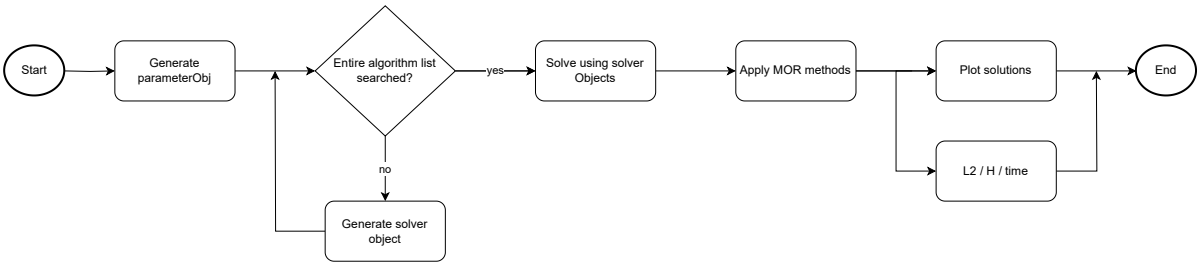


Figure 4.2.: Flow chart of main class

The first step is to generate a parameter object. The parameter object stores all parameters in order to increase the transparency and robustness of the program. The second step is to iterate the array of stated algorithms to solve the heat equation. The options are to solve the heat equation using finite element method 2.2 or using Fourier transform 2.1. After all solver objects have been generated, the according solutions are being computed. After that, the MOR methods are displayed. The final step is to display the solutions or return the $L2$, H_∞ error of each ROM. The processing time can be returned as well.

4.2. Class solution

The solution class is mainly responsible for storing solutions with some meta data. This meta data consists of the method used to generate the stored solution, its predecessor and computing time. The predecessor is important to plot reduced order model solutions side by side with the original finite element solution. The processing time is stored in an array, that typically stores ten measurements.

4.3. Class parameters

The parameters class stores all parameters that have to be distributed. They are all encapsulated within this class, to make distribution and modification easier.

4.4. Class container

Container functions like an abstract class. It implements some methods, such as *euler()* to avoid redundancy within it's child classes. This method takes some Matrices and a vector of initial conditions and generates an solution object, that stores the solution generated using an euler scheme. In case the solution to a ROM is computed, the solution itself has to be transformed. This step is also done here. For this class no objects are invoked.

Algorithm 1 Solve system of ODEs using euler scheme

Input

A matrix of system
B matrix of system
T Coordinate transform to retrieve full state approximation of solution
c_0 Initial condition

Output

solution object

```
1:  procedure EULER
2:      C ← [T c_0]
3:      j ← 2
4:      for t = dt to T do
5:          c_n ← dtAc_0 + dtBh(:, j) + c_0
6:          c_0 ← c_n
7:          C(:, j) ← T c_0
8:          j = j + 1
9:      end for
10:     sol = solution(C, "", 0, 0)
11: end procedure
```

Here dt, T and h are provided by the parameter object, dt is the time step, T is the time limit to which the solution is to be computed and h is the input to the system. Since this method is common among all classes derived from the container class, only the solution is stored in the solution object. The remaining fields of that object are set in the child class after the object has been returned. Those classes derived form container can be seen on 4.3.

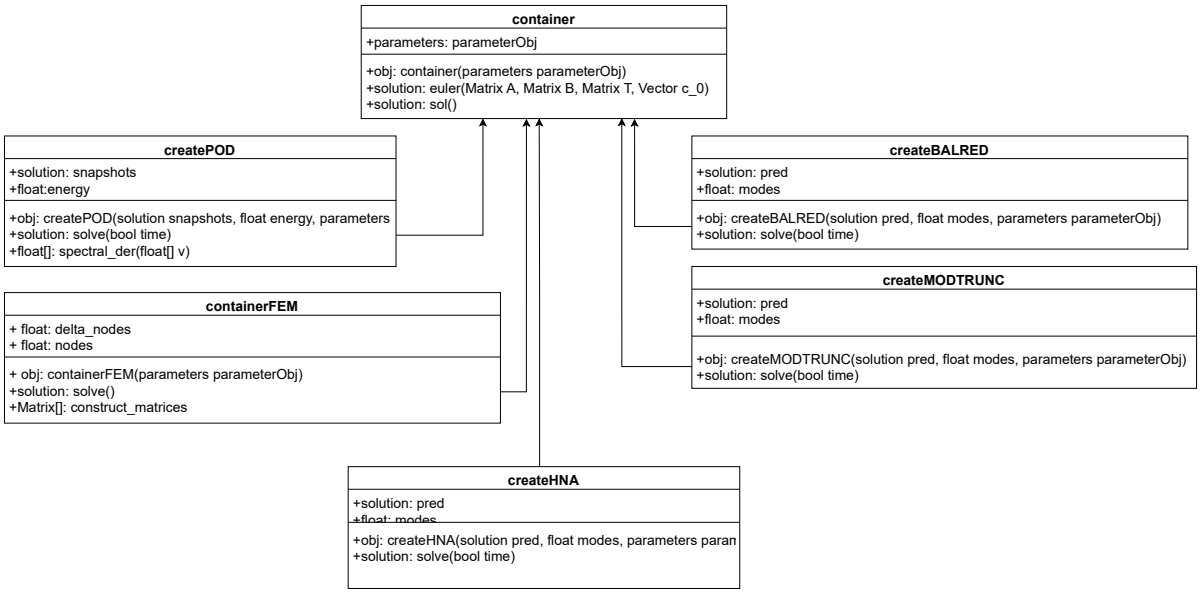


Figure 4.3.: Class diagram of MOR and FEM implementation

4.5. Class createBALRED, createMODTRUNC and createHNA

Those classes are fairly simple. The implementation of the actual model order reduction is done by MORLAB. Hence the according function is called provided with the right set of parameters and the ROM gets generated. After that a solution is generated using the previously described *euler()* method. For time measurements the ROM gets generated ten times and for each cycle the time is stopped. To save time no solutions are generated in this case.

4.6. Class containerFEM

The class containerFEM is responsible for generating a solution and the state space representation of the system using finite element method. FEM is implemented as shown on 4.4.

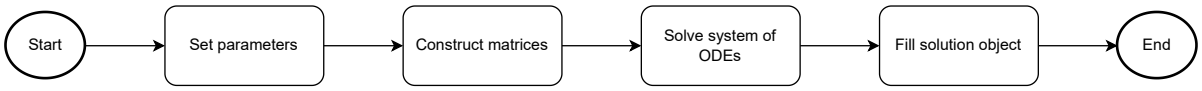


Figure 4.4.: Flow chart for FEM class

The first step is to set the parameters. After that the matrices discussed in 2.2 are being constructed. The next step is to solve the resulting system of ODEs and pass the solution to a solution object.

4.6.1. Construct Matrices

As defined in 2.34 the matrices K and M have to be constructed. This is done by the following method:

Algorithm 2 Construct matrices K and M

Input

Output

[K, M]

procedure ConstructMatrices
  $ii \leftarrow \frac{2}{3}\Delta nodes$
  $ij \leftarrow \frac{1}{6}\Delta nodes$
  $F \leftarrow zeros(nodes, 1)$
  $M \leftarrow zeros(nodes)$
  for $i = 1$ **to** $nodes$ **do**
    $K(i,i) \leftarrow \frac{-2}{\Delta nodes}$
    $M(i,i) \leftarrow ii$
    if $i - 1 > 1$ **then**
      $K(i,i-1) \leftarrow \frac{1}{\Delta nodes}$
      $M(i,i-1) \leftarrow ij$
    end if
    if $i + 1 < nodes + 1$ **then**
      $K(i,i+1) \leftarrow \frac{1}{\Delta nodes}$
      $M(i,i+1) \leftarrow ij$
    end if
  end for
  return$[K, M]$
end procedure

4.6.2. Solve System of ODEs

The resulting system of ODEs is solved using the implementation of the euler scheme in the container class. After the solution is generated, the returned solution object gets filled with the according meta data and returned.

4.7. Class createPOD

The implementation of proper orthogonal decomposition is not included in the use MORLAB package. Therefore it is implemented in the createPOD class. The first step is to construct the matrx Phi as described in 3.2. This is done by computing the SVD of the provided FEM solution and truncating the resulting U matrix, such that the r most dominant modes are included. In the following step the transformed initial condition and the spacial derivative of Phi is computed. The spacial derivative is computed using the spectral derivative. This method to approximate the derivative is chosen over approximations using the difference quotient since it yields better results, especially at the fist and last entry of each column in Φ. In the last step the resulting system of ODEs is solved using the implemented euler scheme and the resulting solution object is filled.

Algorithm 3 Create POD

Input

X Snapshot matrix
time boolean

Output

solution object

1: **procedure** SOLVE
2: rom_time ← []
3: **if** t **then**ime
4: **for** i = 1 **to** 10 **do**
5: $T1$ = tic
6: $[U, S, V] \leftarrow SVD(X, "econ")$
7: Φ = $U(:, 1:r)$
8: $\Phi_{xx} \leftarrow []$
9: **for** i = 1 **to** r **do**
10: $\Phi_{xx}(:, i) \leftarrow spectral\_der(\Phi(:, i))$
11: **end for**
12: rom_time(i) ← $T1 - T2$
13: **end for**
14: sol ← solution(NaN, "POD", X, NaN)
15: sol.rom_time ← rom_time
16: **end if**
17: $[U, S, V] \leftarrow SVD(X, "econ")$
18: Φ = $U(:, 1:r)$
19: $\Phi_{xx} \leftarrow []$
20: **for** i = 1 **to** r **do**
21: $\Phi_{xx}(:, i) \leftarrow spectral\_der(\Phi(:, i))$
22: **end for**
23: $a_0 \leftarrow \Phi^T u_0$
24: $sol \leftarrow euler(\alpha\Phi^T\Phi_{xx}, \Phi^T, \phi, a0)$
25: rom = struct("A", $\alpha\Phi^T\Phi_{xx}$, "B",Φ^T, "C", Φ, "D", parameterObj.D);
26: sol.method = "Proper Orthogonal Decomposition"
27: sol.pred = X
28: sol.reduced_model = rom
29: **end procedure**

From line three to fifteen there obviously some redundancy. This section is mend to measure the processing time of POD without generating a solution. The time measurements are stored in a solution object. The spectral derivative is implemented in the following way.

Algorithm 4 Compute spectral derivative

Input

v Vector **Output**

v' Derivative of v

1: **procedure** SPECTRAL_DER()

2: $v' \leftarrow -k^2$ fft(v)

3: **end procedure**

Here k are the wave numbers and all multiplication is element wise multiplication.

4.8. Measurements

In the next chapter some measurements are taken to gain information about the performance and error of each method. Therefore three sets measurements are taken. The first two are the $L2$ and H_∞ error of the results and the last one is the processing time. Further details are in the according chapter.

4.8.1. L2 error

The $L2$ error gets determined in the *compare()* method in the main class. It get computed in the following way.

$$E2 = ||S - S'||_F \tag{4.1}$$

Here S is the solution of the ROM, S' is the original FEM solution.

4.8.2. H_∞ error

The H_∞ error also gets determined in the *compare()* method in the main class.

$$H = ||S - S'||_{H_\infty} \tag{4.2}$$

Here S is the transfer function of the ROM, S' is the transfer function of original FEM system.

4.8.3. Processing time

As already mentioned the processing time gets determined in the according container class. It gets passed through the *compare()* method in the main class.

5. Comparison of MOR Methods

The previously mentioned methods for model order reduction will be compared regarding time domain error, frequency domain error and computational speed. The time domain error will be obtained by comparing the FEM solution to a given approximation. To get insights into the frequency domain error, the error system of a reduced order model will be analysed. The computational speed will be determined by measuring the time it takes to generate a ROM. Here it is assumed that the implementations provided by MORLAB are programmed in a sufficiently effective manner. For testing the following parameters are used

$$\alpha = 0.1, \quad T = 1, \quad L = 1 \tag{5.1}$$

$$n = 100, \quad n_t = 10^4 \tag{5.2}$$

.
These values are chosen such that it yields results in a timely manner. Especially n_t and α are important for stability of euler scheme. If both are too low, the euler scheme becomes unstable. Choosing n_t too high the time and memory consumption becomes rather large. There was no exact method for determining the parameters in this way.

5.1. Time Domain Error

The time domain error will defined as $\epsilon = Y - \hat{Y}$ where Y and $\hat{Y}$ denote the matrices storing the output of the systems G and G_r. Since the output matrices are usually rather large, it is impractical to use ϵ directly. Therefore $\|\epsilon\|_F$ will be considered. The error is measured using the Frobenius norm to get a measure of the error that respects all data points. Here two aspects are interesting. There will be two different initial conditions considered. The first one is $x(0, x) = 1$. It is chosen in this way to display the workings of the boundary condition. If there were no boundary conditions, for a constant initial condition the system would never cool down since $\frac{\partial^2 u}{\partial x^2} = 0$ (2.6) at every point in time. This only holds if there is no input to the system. Therefore $u(t) = 0$, where u denotes the systems input. The second initial condition is $x(0, x) = \sin(\frac{2\pi}{L}x)$ with random input. The initial condition is sinusoidal since it is easy to approximate.

5.1.1. Proper Orthogonal decomposition

Figure 5.1 shows the $L2$ error of the solution obtained by the POD model.

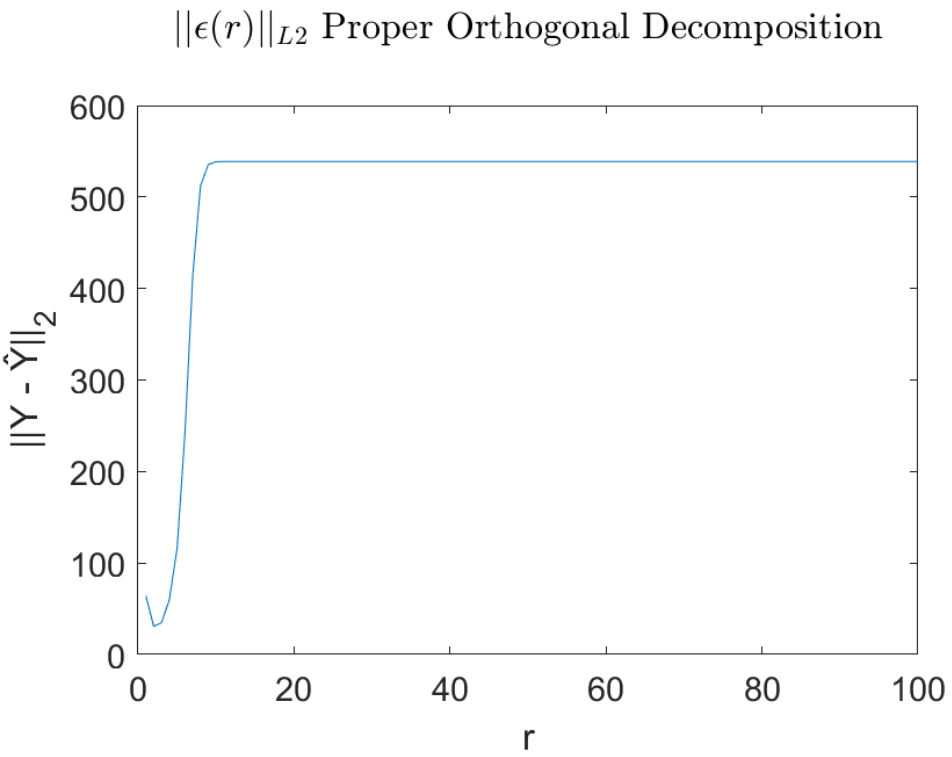


Figure 5.1.: L2 Error Proper Orthogonal Decomposition $x(0, x) = 1$

It is clearly visible that the error is minimal at $r = 2$. This is to be expected since for large r also low variance modes will be included. Figure 5.2 shows that the first two modes capture the most variance with roughly 96% combined. It is also visible that all the remaining modes combined only capture 4%.

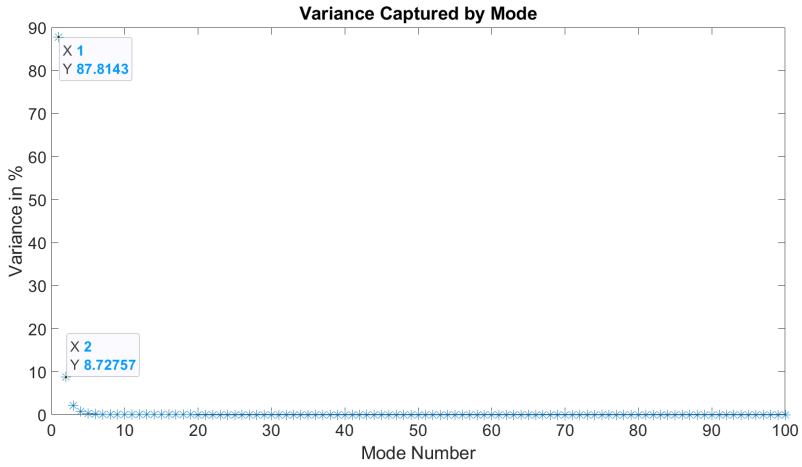
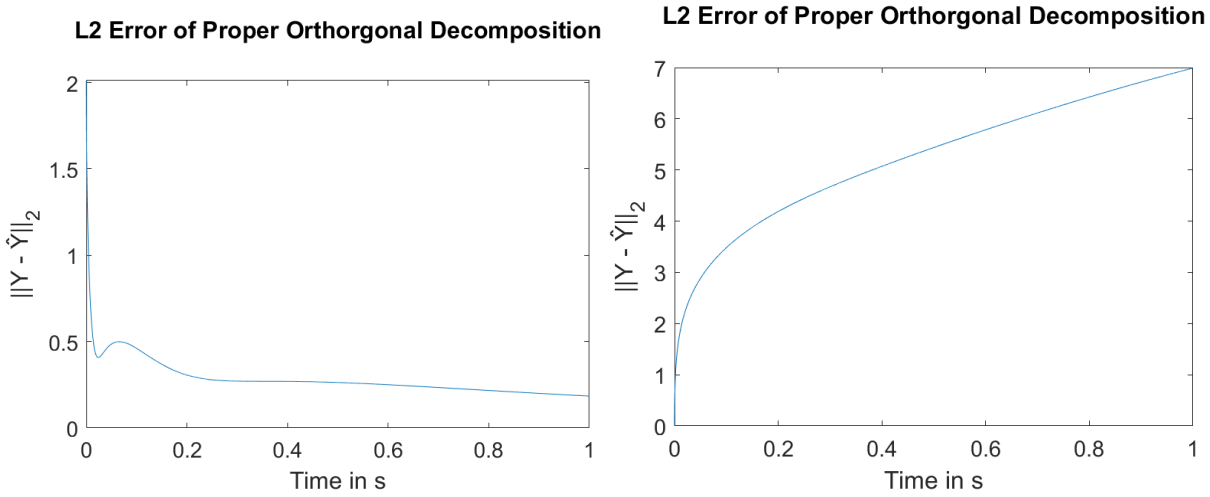


Figure 5.2.: Variance of POD Modes

In this case the low variance modes introduce error, since the initial condition is poorly approximated using orthogonal basis vectors. This can be seen on figure ??.



(a) L2 POD error $r = 2$, $n = 100$, $x(0, x) = 1$ (b) L2 POD error $r = 100$, $n = 100$, $x(0, x) = 1$

The $L2$ error for $r = 2$ can be seen on 5.3a. Here for $t = 0$ the error is the largest and then drops of quickly and seems to converge to roughly 0.3 whereas the error on 5.3b for $r = 100$ is almost zero at $t = 0$ and diverges. This problem does not only apply to the initial condition, this problem if the snapshot matrix contains vectors that are poorly approximated using orthogonal basis vectors. The low variance modes do not introduce error. In fact they even lower the error and the error is lower in general as it can be seen on figure 5.4.

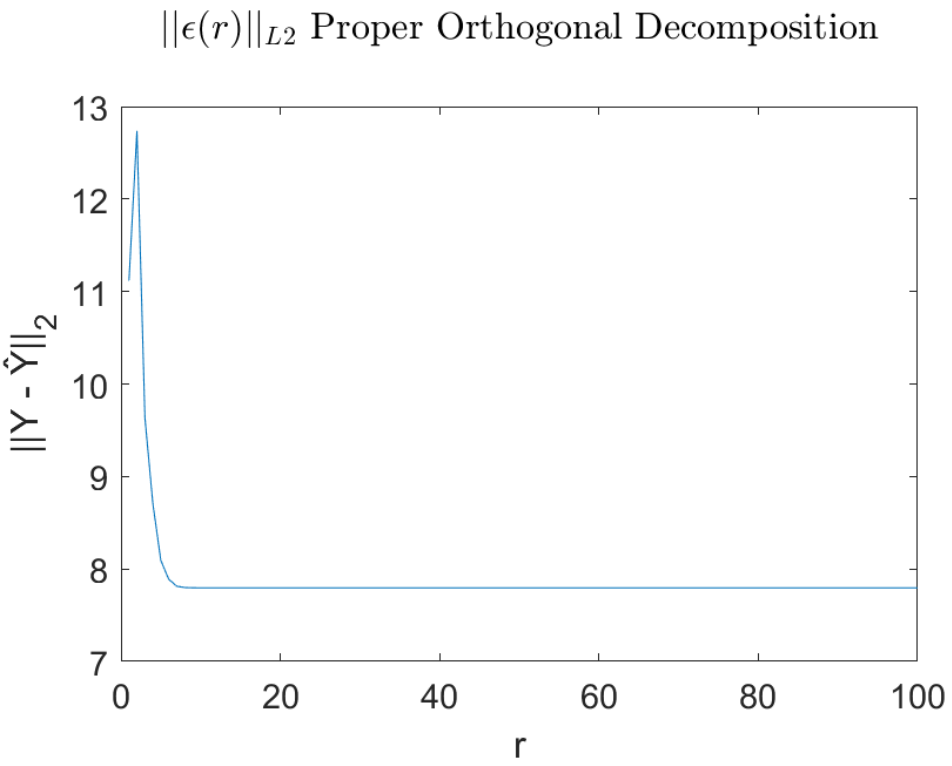
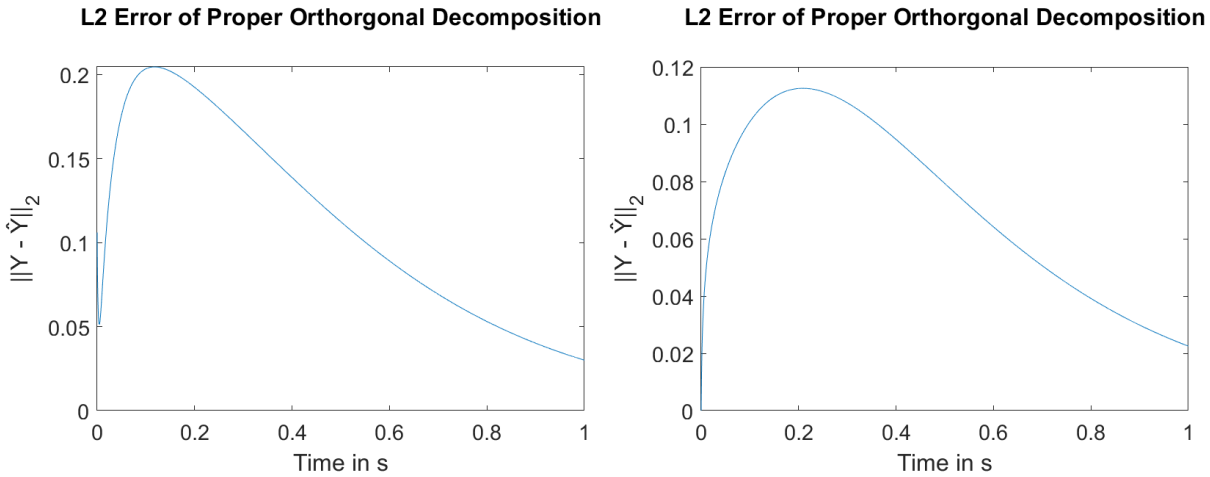


Figure 5.4.: L2 Error Proper Orthogonal Decomposition $x(0,x) = \sin(\frac{2\pi}{L}x)$

Another difference is that for $x(0,x) = \sin(\frac{2\pi}{L}x)$ the error distribution over time does not differ as much for $r = 100$ and $r = 2$. This can be observed on figure 5.5a and 5.5b.

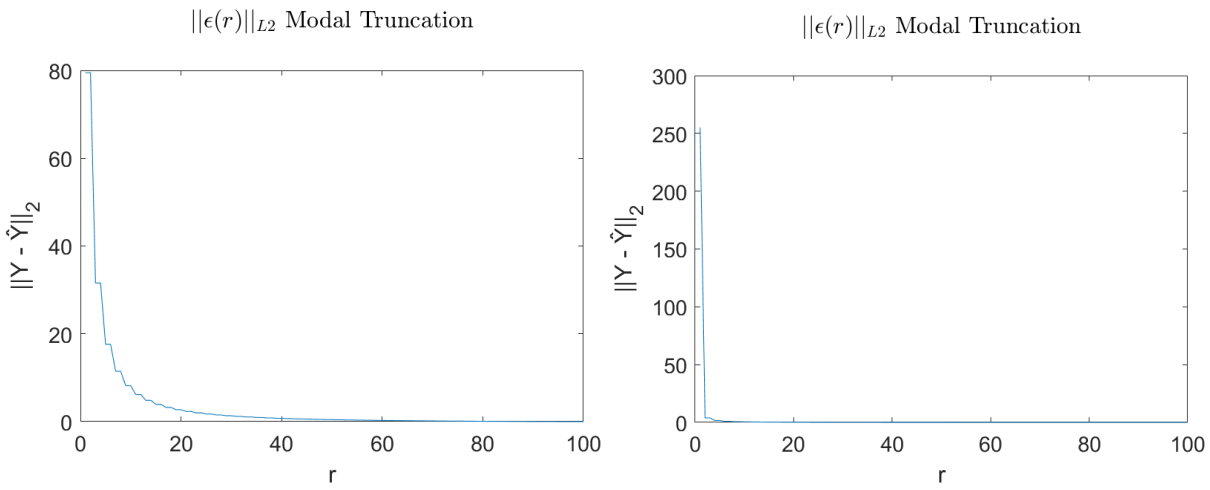


(a) $r = 2$, $n = 100$, $x(0,x) = \sin(\frac{2\pi}{L}x)$ (b) $r = 100$, $n = 100$, $x(0,x) = \sin(\frac{2\pi}{L}x)$

Figure 5.5.: L2 POD error

5.1.2. Modal Truncation

Figure 5.6a shows the $L2$ error of the ROM obtained by Modal Truncation.



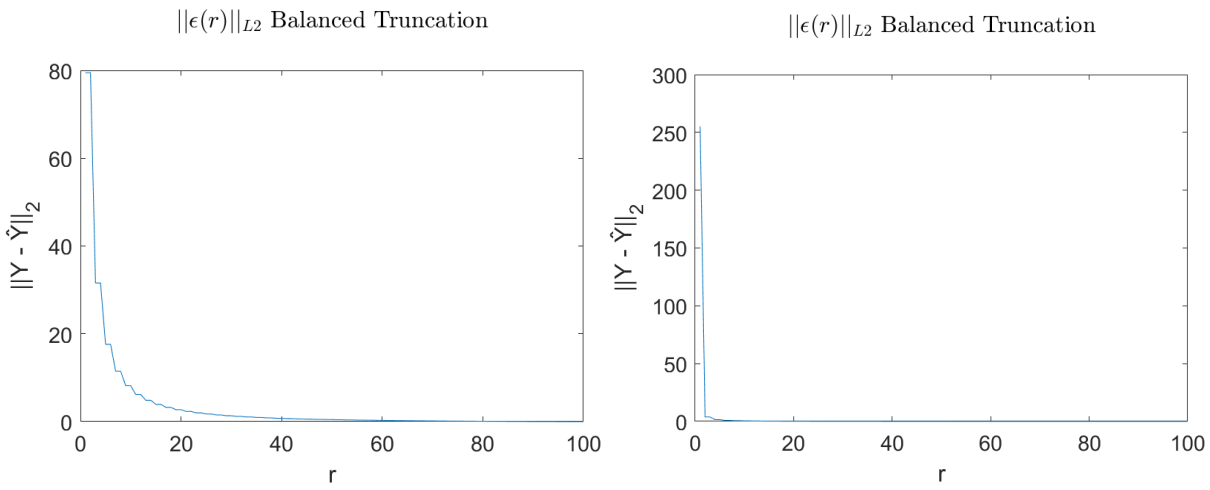
(a) $x(0,x) = 1$ (b) $x(0,x) = \sin(\frac{2\pi}{L}x)$

Figure 5.6.: L2 Error Balanced Truncation

It shows that as r increases the error converges to zero. This is to be expected since as r increases G_r becomes closer to G. For $x(0,x) = \sin(\frac{2\pi}{L}x)$ the error starts off significantly higher but decays even faster as it can be seen on figure 5.6b.

5.1.3. Balanced Truncation

Figure 5.7a shows the $L2$ error of the ROM obtained by Balanced Truncation.



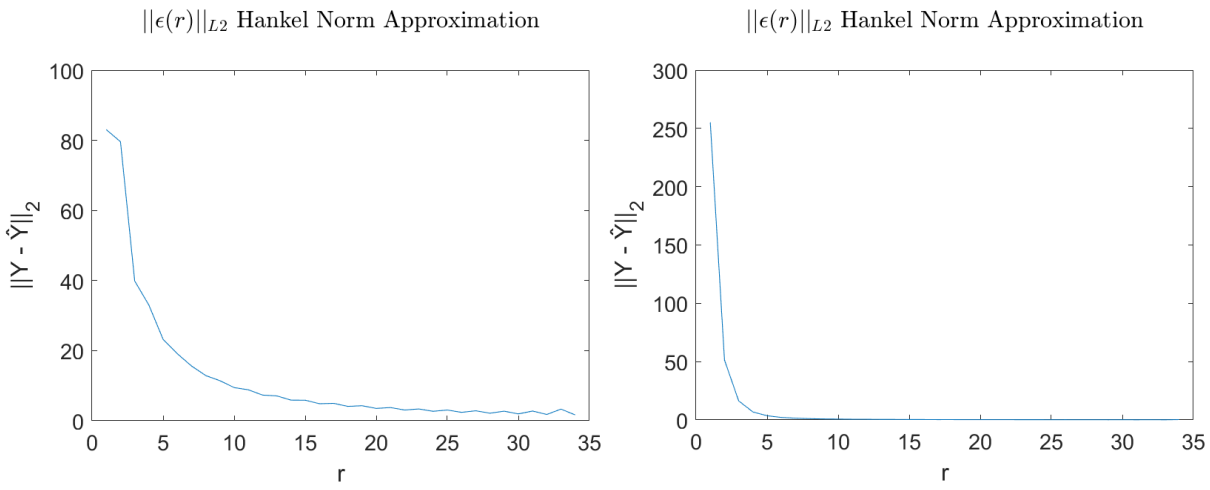
(a) $x(0,x) = 1$ (b) $x(0,x) = \sin(\frac{2\pi}{L}x)$

Figure 5.7.: L2 Error Balanced Truncation

It shows that as r increases the error converges to zero. This is to be expected since as r increases G_r becomes closer to G. As expected the results for Modal Truncation and Balanced Truncation are the same.

5.1.4. Hankel Norm Approximation

Figure 5.8a shows the $l2$ error of the output generated by the HNA model. Here it is striking that for $r > 34$ the model does not generate usable output. This is due to the reason that for the chosen settings the used euler scheme does not converge for $r > 34$. Decreasing Δt yields results for $r > 34$ but it becomes prohibitively expensive to compute. However for $r \leq \frac{n}{3}$ the error converges to zero as r increases.



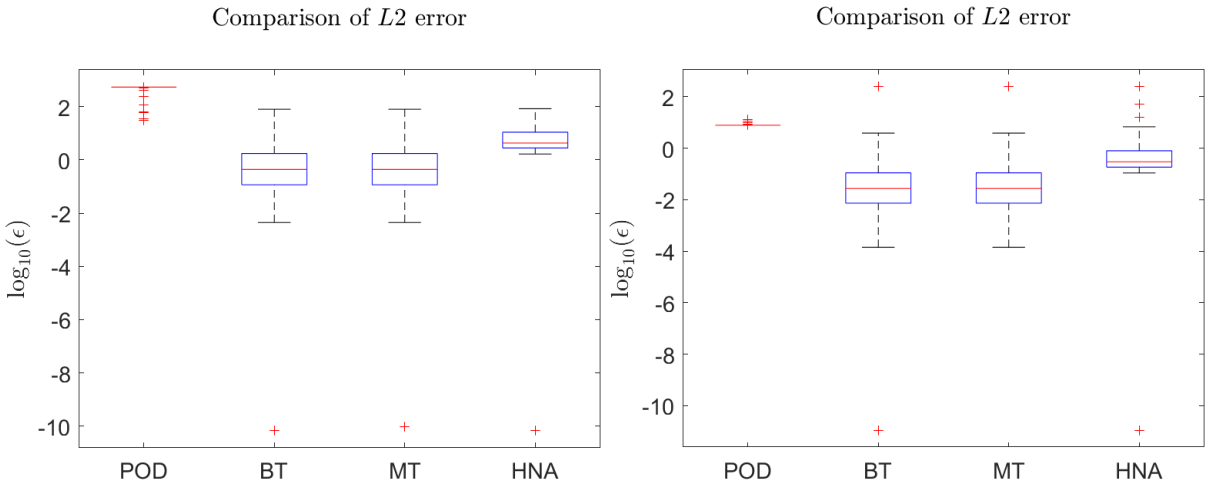
(a) $x(0,x) = 1$ (b) $x(0,x) = \sin(\frac{2\pi}{L}x)$

Figure 5.8.: $L2$ Error Hankel Norm Approximation

It is noticeable that similar to BT and MT the error for $x(0,x) = \sin(\frac{2\pi}{L}x)$ starts higher than for $x(0,x) = 1$ and then decays quite fast.

5.1.5. Comparison of Time Domain Error

Figure 5.9 shows the measured $L2$ error plotted as a box plot to compare them. Since the range of the data is too large to plot it in this way, the data is transformed using $\log_{10}$. This results in the problem that zeros cannot be expressed. It is the case for Modal Truncation at $r = 100$.



(a) $x(0,x) = 1$ (b) $x(0,x) = \sin(\frac{2\pi}{L}x)$

Figure 5.9.: Box Plot of $L2$ Error

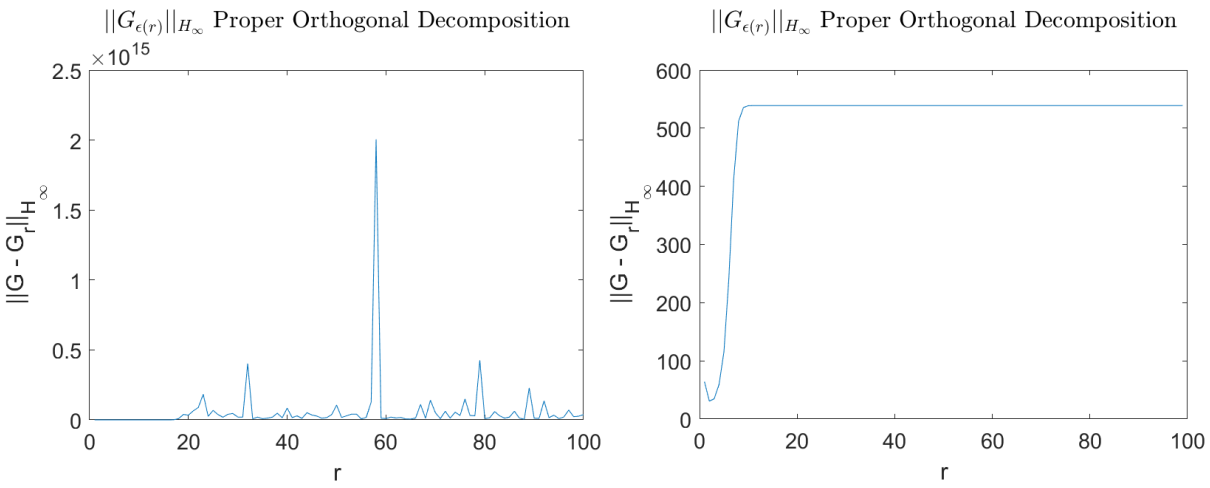
It is noticeable that the relative difference between each of the methods stays roughly the same for both initial conditions. As expected the error of all methods is higher in general for $x(0,t) = 1$. The box for POD is the one with the lowest IQR while having the largest $\log$ error, meaning that the error of POD converges the fastest but also it does not converge to zero. Keep in mind that the seemingly large IQR of the error of MT and BT is actually small, because the $\log$ error is depicted. Therefore the error of MT and BT converge faster than the one of HNA while the error is smaller in general. Therefore BT and MT are the best choices when considering $L2$ time domain error.

5.2. Frequency Domain Error

The frequency domain error for $G_\epsilon = G - G_r$ will be determined using the H_∞ norm. For a MIMO system this is the maximum singular value of the frequency response of the transfer-matrix. It can be interpreted as the maximum gain from the i^{th} input to the i^{th} output [37]. The $H2$ cannot be used to get a measure of the error since the ROM that HNA yields has nonzero feedthrough matrix, therefore $||G_\epsilon||_{H_2}$ is infinite.

5.2.1. Proper Orthogonal Decomposition

Since the ROM obtained by POD is derived from a solution of the full state system, it is depended on the initial condition. Therefore the frequency domain error is also dependent on the initial condition. This can be seen on figure 5.10a and figure 5.10b, they clearly differ from each other. For $x(0,x) = 1$ there are two distinct spikes at $r = 49$ and $r = 63$. The reason for them is unclear, however the larger one is three orders of magnitude higher than the peak in the H_∞ error for $x(0,x) = \sin(\frac{2\pi}{L}x)$. This peak is not depicted on 5.10b since it is one single data point that is 14 orders of magnitude higher than the remaining data points. The peaks on 5.10a are not removed since the remaining data points are also large.



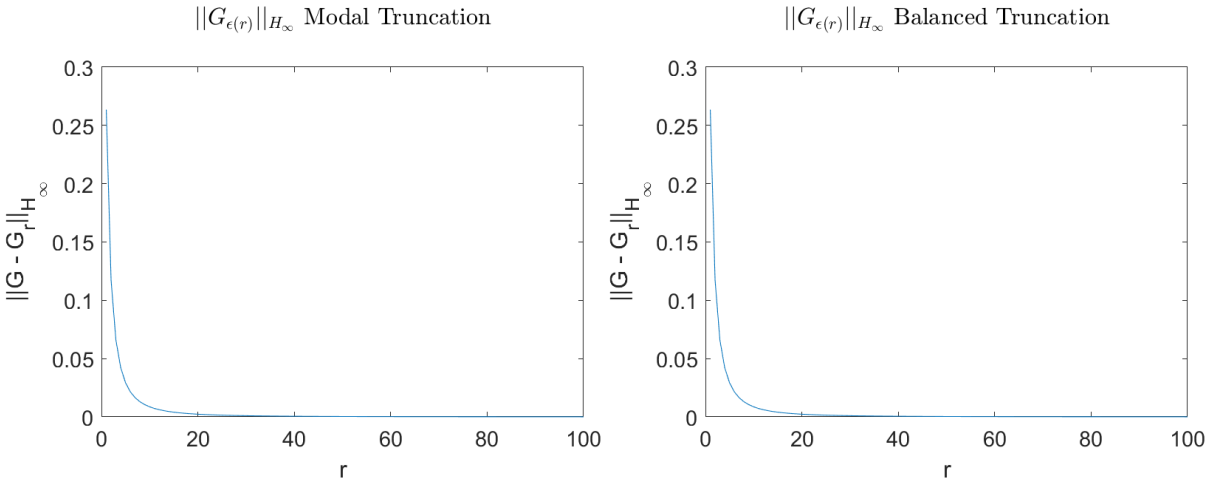
(a) H_∞ error of POD, $x(0,x) = 1$ (b) H_∞ error of POD, $x(0,x) = \sin(\frac{2\pi}{L}x)$

Figure 5.10.: H_∞ error of POD

In general the error in frequency domain is larger for $x(0,x) = 1$ than for $x(0,x) = \sin(\frac{2\pi}{L}x)$. This matches the behaviour of the time domain error.

5.2.2. Modal Truncation and Balanced Truncation

The frequency domain error of Modal Truncation and Balanced Truncation is independent of the initial condition. For small r it is comparably large and converges towards zero as r gets larger. As expected the results of Balanced Truncation and Modal Truncation is the same as figure 5.11 shows.



(a) H_∞ error of Modal Truncation (b) H_∞ error of Balanced Truncation

Figure 5.11.: H_∞ error of Balanced Truncation and Modal Truncation

5.2.3. Hankel Norm Approximation

Figure 5.12 shows the frequency domain error of HNA. Similar to BT and MT the error is large for small r and decays as r gets larger. However the error overall is smaller than for BT and HNA.

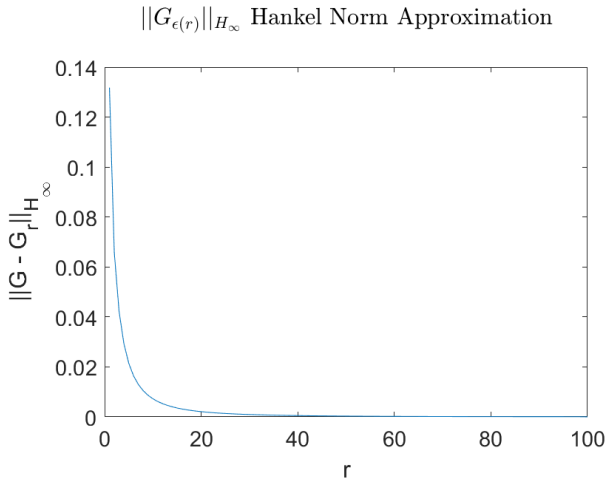


Figure 5.12.: H_∞ error of Hankel Norm Approximation

5.2.4. Comparison of Time Domain Error

To compare the results directly, the error of the ROMs for all r are plotted using a box plot. However the error is not directly used because of some large peaks in the data, which would render the plot unreadable. Instead the $\log_{10}$ of each data point is taken. This is possible since all data points are strictly positive, therefore the order of the data points is preserved while lowering the range between outliers. The box plot can be seen on 5.13.

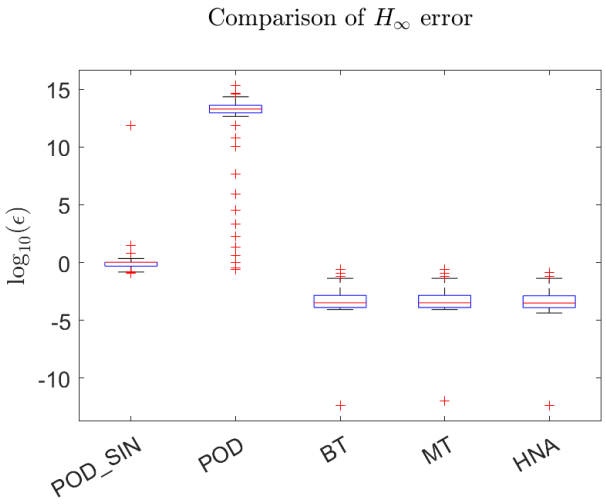


Figure 5.13.: Box Plot of H_∞ error

Where 'POD' denotes the POD ROM for $x(0,x) = 1$ and 'POD_SIN' denotes the POD ROM for $x(0,x) = \sin(\frac{2\pi}{L}x)$. The plot shows that the frequency domain error of the ROM obtained using POD with $x(0,x) = 1$ yields by far the worst results. The best choice for minimal H_∞ error here is clearly HNA. The error of HNA is significantly lower than the error of the other three methods.

5.3. Processing Time

The processing time is obtained by measuring the time it takes to build the reduced order model. To make sure that the measurements are reliable to some degree, each measurement is taken ten times. This choice is arbitrary, but was taken to keep overall computing time in check. Those measurements are depicted as box plots in the following figures.

5.3.1. Proper Orthogonal Decomposition

Figure 5.14 shows the processing time of POD. It can be observed that the time it takes to compute a POD model, is quite constant for $1 \geq r \geq 100$. This implies that the implementation of the algorithm is efficient to an extend that the range of r is too small to gain information about the relation between r and the processing time. The measurements are mainly between 0.012 and 0.022 seconds.

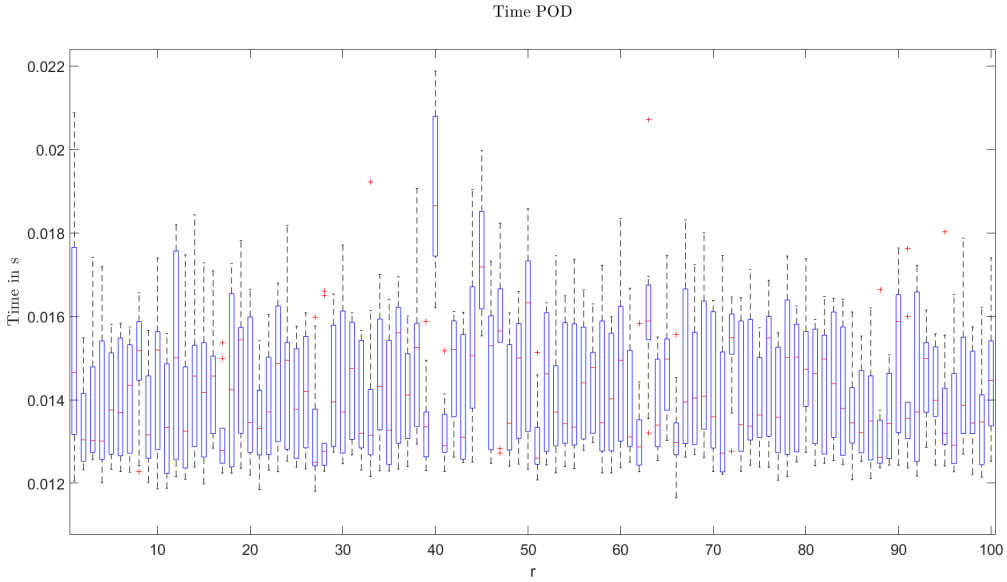


Figure 5.14.: Processing time POD

5.3.2. Modal Truncation

Figure 5.15 shows the processing time of modal truncation. It is noticeable that $r = 100$ yields the lowest time to compute. This is to be expected since in this case modal truncation is equal to applying a coordinate transform to the original system. For $r \neq 100$ there is seemingly some r around fifty that offers some optimal computing speed. Again this could be due to the rather small range of r. The most measurements are in a range between 3.5 and 5 milliseconds.

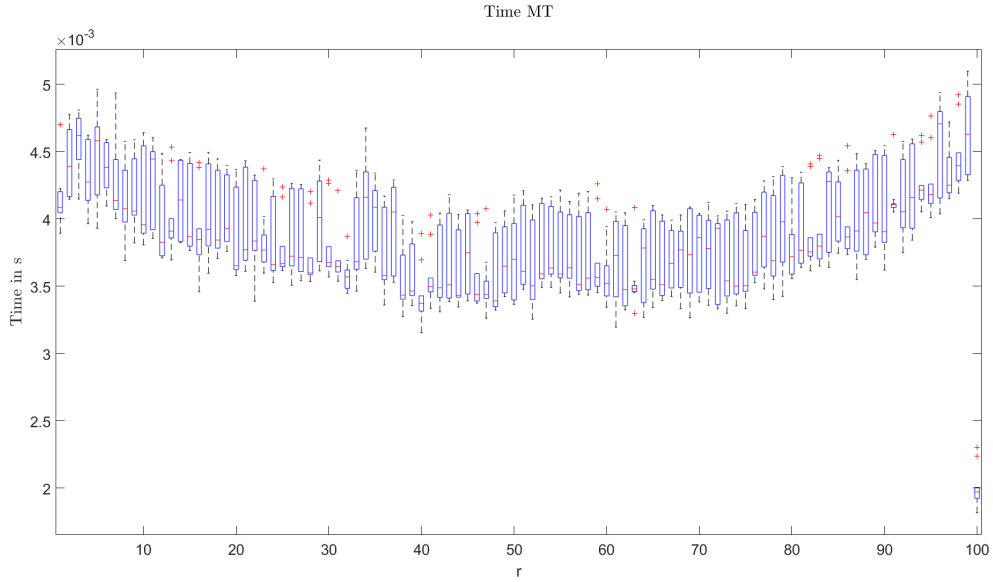


Figure 5.15.: Processing time MT

5.3.3. Balanced Truncation

Figure 5.16 displays the processing time with respect to the model order r. Again it is fairly constant where the majority of measurements is in a range between 0.03 and 0.04 seconds.

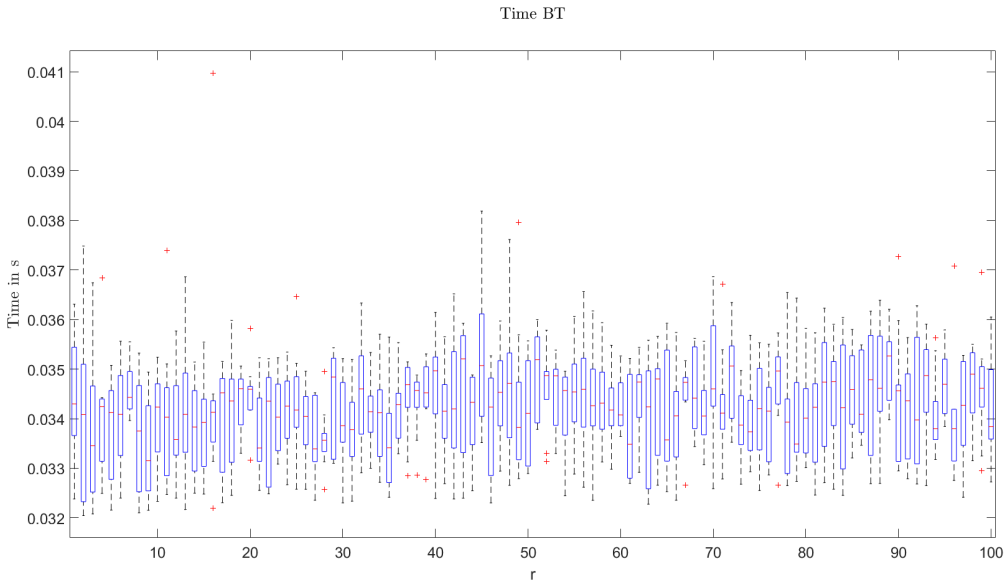


Figure 5.16.: Processing time BT

5.3.4. Hankel Norm Approximation

Figure 5.17 shows the the processing time dependent on the model order. It is striking that for $r = 99$ and $r = 100$ the processing time is significantly lower than in the remaining measurements. The exact reason for this is unclear, however it could be speculated that for $r = 100$ only a balanced realization is computed and therefore many steps of the algorithm could be skipped. Except for those two outliers the range of the majority of the measurements is 0.036 to 0.043 seconds.

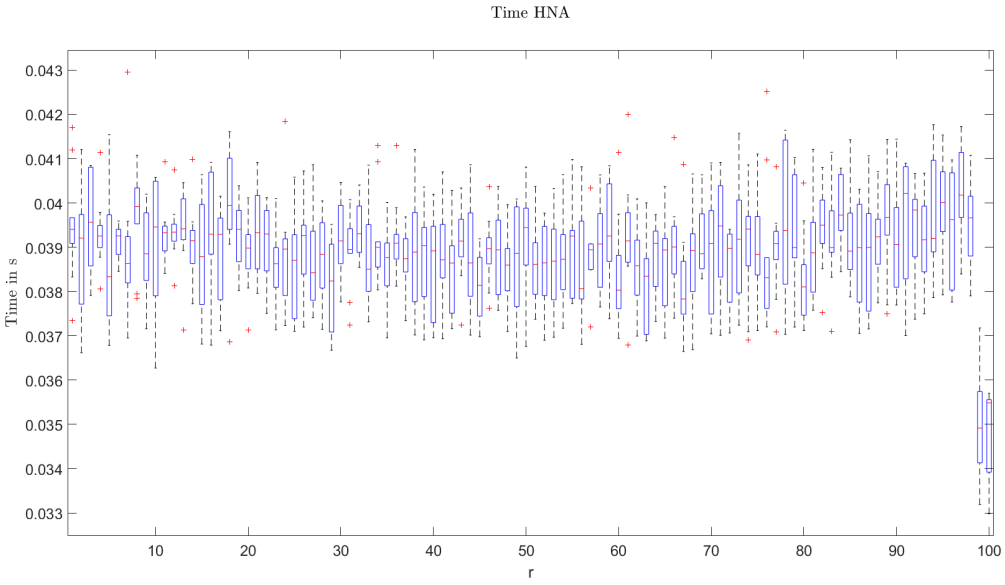


Figure 5.17.: Processing time HNA

5.3.5. Comparison of processing time

The fastest method here is modal truncation. It is roughly one order of magnitude faster than the remaining methods. Balanced truncation and hankel norm approximation are quite similar in processing time, however hankel norm approximation is slightly slower. The slowest method is hankel norm approximation. This is not surprising since POD uses a naive implementation, no toolbox is used. So the best method when it comes to processing speed is modal truncation.

6. Conclusion and Outlook

6.1. Conclusion

In summary, the basics were first described. First, the 1D heat equation was explained and a naïve approach to solving it was presented. This approach had the problem that no boundary conditions could be met. For this reason, the finite element method was introduced and then it was described how the heat equation can be discretised and solved with this method. Subsequently, various fundamentals were dealt with that were necessary for understanding the later procedures for model order reduction. These were explained in the following chapter. The orthogonal decomposition, the modal truncation, the balanced truncation and the hankel norm approximation were used. It is striking that the modal truncation and the balanced truncation are identical for the example system resulting from the heat equation. In the following chapter, in which the presented methods were compared in terms of $L2$ error, H_∞ error and running time, this finding was confirmed. The errors of the modal truncation and the balanced truncation were identical. It is also noticeable that both the $L2$ error and the H_∞ error for POD depend strongly on the initial values of the system used to create the snapshot matrix. This can be explained by the fact that constant initial values in particular are difficult to approximate with a small number of orthogonal basis vectors, while sinusoidal initial values are very easy to approximate. In comparison, balanced truncation and modal truncation are the best in terms of $L2$ and H_∞ error. Hankel norm approximation is only slightly worse here. POD was by far the worst, especially if the snapshot matrix was created with constant initial values. In terms of computing time, balanced truncation and hankel norm approximation differ only slightly. Proper orthogonal decomposition is about a factor of 2 faster here. A rather naive approach was chosen for the implementation of the POD algorithm, without any special optimisation of the performance. This speaks for the simplicity of the method. Modal truncation performed best. It was about an order of magnitude faster than balanced truncation or hankel norm approximation. Thus it can be said that modal truncation is the best method, in terms of computing time and error, for model order reduction for the system described.

6.2. Outlook

It would be worthwhile to investigate how the selected methods behave for more complex systems than the 1D heat equation. Possible systems would be 3D heat equations for inhomogeneous materials or, for example, 3D Navier-Stokes equations for the simulation of fluids. It would be interesting to choose the system in such a way that the system cannot be arbitrarily influenced from the outside and that not every state can be measured perfectly, as this has led to modal truncation and balanced truncation giving the same results. It would also make sense to use larger initial systems for the comparisons, since it has been shown

that, especially in the time measurement, hardly any significant statements have been made regarding the runtime as a function of the model order.

Literaturverzeichnis

[1] Gustafsson, B. "Heat Conduction". In: *Fundamentals of Scientific Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 255–262. URL: https://doi.org/10.1007/978-3-642-19495-5_16.

[2] Papula, L. "Gewöhnliche Differentialgleichungen". In: *Mathematik für Ingenieure und Naturwissenschaftler Band 2: Ein Lehr- und Arbeitsbuch für das Grundstudium*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, pp. 343–541. URL: https://doi.org/10.1007/978-3-658-07790-7_4.

[3] Gustafsson, B. "Finite Element Methods". In: *Fundamentals of Scientific Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 173–192. URL: https://doi.org/10.1007/978-3-642-19495-5_11.

[4] Gustafsson, B. "Polynomial Expansions". In: *Fundamentals of Scientific Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 117–123. URL: https://doi.org/10.1007/978-3-642-19495-5_7.

[5] Gustafsson, B. "Least Square Problems". In: *Fundamentals of Scientific Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 125–133. URL: https://doi.org/10.1007/978-3-642-19495-5_8.

[6] Brunton, S. L./ Kutz, J. N. "Singular Value Decomposition (SVD)". In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 3–46.

[7] Brunton, S. L./ Kutz, J. N. "Fourier and Wavelet Transforms". In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 47–83.

[8] Szabo, F. E. "U". In: *The Linear Algebra Survival Guide*. Ed. by Szabo, F. E. Boston: Academic Press, 2015, pp. 385–392. URL: https://www.sciencedirect.com/science/article/pii/B978012409520550028X.

[9] Naoko, S. *Geometric Interpretation of the Correlation between Two Variables*. 2018. URL: https://medium.com/@ns2586/geometric-interpretation-of-the-correlation-between-two-variables-4011fb3ea18e (visited on 03/01/2023).

[10] Oliver, K. *Math 19b: Linear Algebra with Probability*. 2011. URL: https://people.math.harvard.edu/%5C~knill/teaching/math19b_2011/handouts/lecture12.pdf (visited on 03/01/2023).

[11] Victor, L. *IAML: Dimensionality Reduc6on*. 2011. URL: https://www.inf.ed.ac.uk/teaching/courses/iaml/2011/slides/pca.pdf (visited on 03/01/2023).

[12] Douglas, B. "The Fundamentals of Control Theory". In: Academia, 2019, pp. 8–15.

[13] Douglas, B. "The Fundamentals of Control Theory". In: Academia, 2019, pp. 42–47.

[14] Benner, P. et al. "1 Model order reduction: basic concepts and notation". In: *Volume 1 System- and Data-Driven Methods and Algorithms*. Ed. by Benner, P. et al. Berlin, Boston: De Gruyter, 2021, pp. 1–14. URL: https://doi.org/10.1515/9783110498967-001.

[15] Brunton, S. L./ Kutz, J. N. "Linear time-invariant systems". In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, p. 334.

[16] Douglas, B. "The Fundamentals of Control Theory". In: Academia, 2019, pp. 48–63.

[17] Abell, M. L./ Braselton, J. P. "Chapter 8 - Introduction to the Laplace Transform". In: *Introductory Differential Equations (Fifth Edition)*. Ed. by Abell, M. L./ Braselton, J. P. Fifth Edition. Academic Press, 2018, pp. 399–460. URL: https://www.sciencedirect.com/science/article/pii/B9780128149485000082.

[18] Brunton, S. L./ Kutz, J. N. "CONTROLLABILITY AND OBSERVABILITY". In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 335–341.

[19] *Volume 1 System- and Data-Driven Methods and Algorithms*. Berlin, Boston: De Gruyter, 2021. URL: https://doi.org/10.1515/9783110498967.

[20] Gustafsson, B. "Polynomial Expansions". In: *Fundamentals of Scientific Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 117–123. URL: https://doi.org/10.1007/978-3-642-19495-5_7.

[21] Gustafsson, B. “Numerical Methods for Differential Equations”. In: *Fundamentals of Scientific Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 137–144. URL: https://doi.org/10.1007/978-3-642-19495-5_9.

[22] Brunton, S. L./ Kutz, J. N. “Reduced Order Models (ROMs)”. In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 375–402.

[23] Brunton, S. L./ Kutz, J. N. “Discrete Fourier transform (DFT) and fast Fourier transform (FFT)”. In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 71–73.

[24] Ruppert, F. *Differentiation mit Matrizen*. matheplanet.com. URL:https://www.matheplanet.com (version: 2014-03-13). eprint: https://www.matheplanet.com/matheplanet/nuke/html/viewtopic.php?rd2\&topic=128338\&start=0\#p937673. URL: https://www.matheplanet.com/matheplanet/nuke/html/viewtopic.php?rd2%5C&topic=128338%5C&start=0%5C#p937673.

[25] Kloos, J. *When is matrix multiplication commutative?* Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/170371 (version: 2014-03-13). eprint: https://math.stackexchange.com/q/170371. URL: https://math.stackexchange.com/q/170371.

[26] Sapir, D. M. V. *Symmetric, diagonal, triangular matrices*. Linear Algebra Web-Notes. URL: https://math.vanderbilt.edu/sapirmv/msapir/jan22.html#:~:text=following%20result%20holds.-,Theorem.,B%20then%20AB%20is%20symmetric..

[27] Brunton, S. L./ Kutz, J. N. “BALANCED MODELS FOR CONTROL”. In: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 376–388.

[28] Vuillemin, P./ Maillard, A./ Poussot-Vassal, C. *Optimal Modal Truncation*. 2020. arXiv: 2009.11540 [math.OC].

[29] Peter Benner, L. F. *Model Reduction for Dynamical Systems — Lecture 4 —*. 2015. URL: https://www.mpi-magdeburg.mpg.de/2925234/lecture4.pdf (visited on 04/03/2023).

[30] Singh, N. “Reduction of linear dynamic systems using hankel norm approximation”. In: *Control and Intelligent Systems* 41 (01/2013).

[31] GLOVER, K. “All optimal Hankel-norm approximations of linear multivariable systems and their L, ∞-error bounds†”. In: *International Journal of Control* 39.6 (1984), pp. 1115–1193. eprint: https://doi.org/10.1080/00207178408933239. URL: https://doi.org/10.1080/00207178408933239.

[32] Sandberg, H. “Introduction to Model Order Reduction”. In: (07/2019), pp. 45–48. URL: http://www.diva-portal.org/smash/get/diva2:1338145/FULLTEXT01.pdf (visited on 04/15/2023).

[33] Zhang, X. *The properties and application of symmetric matrice*. 09/2021. URL: https://towardsdatascience.com/the-properties-and-application-of-symmetric-matrice-1dc3f183de5a (visited on 04/21/2023).

[34] (https://math.stackexchange.com/users/57021/yiorgos-s-smyrlis), Y. S. S. *Integral of matrix exponential*. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/65 (version: 2020-10-28). eprint: https://math.stackexchange.com/q/658289. URL: https://math.stackexchange.com/q/658289.

[35] Benner, P./ Werner, S. W. R. *Morlab*. URL: https://www.mpi-magdeburg.mpg.de/projects/morlab (visited on 02/27/2023).

[36] Systems, P. D. *MOR toolbox*. URL: https://mordigitalsystems.fr/en/products.html (visited on 02/27/2023).

[37] Schuster, E. *System Identification and Robust Control*. 2020. URL: https://www.lehigh.edu/~eus204/teaching/ME450_SIRC/lectures/lecture07.pdf (visited on 04/23/2023).

[38] D‘Erchie, M. e. a. “Analysis Eins”. In: *Mathe für Nicht-Freaks*. Serlo Education e. V., 2017, pp. 111–113. URL: https://mathe-builds.serlo.org/analysis1_final/analysis1_final.pdf.

[39] Alexandre M. Bayen, T. S. “An Introduction to MATLAB® Programming and Numerical Methods for Engineers”. In: Science Direct, 2015, pp. 211–214. URL: https://www.sciencedirect.com/book/9780124202283/an-introduction-to-matlab-programming-and-numerical-methods-for-engineers.

A. Appendix

A.1. Deriving matrices for FEM using piecewise linear functions

The so called triangle function is defined as follows:

$$\phi_j(x) = \begin{cases} (x - x_{j-1})/\Delta x), & x_{j-1} \leq x < x_j \\ (x_{j+1} - x)/\Delta x), & x_j \leq x < x_{j+1} \\ 0, \quad \text{otherwise} \end{cases} \tag{A.1}$$

[4]
The following integrals have to be evaluated:

$$\int_\chi \phi_j \phi_k dx \quad \forall \phi_k \in \phi \tag{A.2}$$

$$-\int_\chi \frac{d\phi_j}{dx}\frac{d\phi_k}{dx} dx \quad \forall \phi_k \in \phi \tag{A.3}$$

With $\chi \subset \mathbb{R}$. Note that the product of two functions ϕ_j and ϕ_k and their derivatives is only under two conditions not zero:

1. $k = j$

Considering this case the integral A.2 becomes:

$$\int_{x_{j-1}}^{x_j} \phi_j^2 dx + \int_{x_j}^{x_{j+1}} \phi_j^2 dx \tag{A.4}$$

Because of symmetry only one of the above integrals have to computed:

$$2\int_{x_{j-1}}^{x_j}\phi_j^2 dx \tag{A.5}$$

$$=\frac{2}{\Delta x^2}\int_{x_{j-1}}^{x_j}(x-x_{j-1})^2 dx \tag{A.6}$$

$$\frac{2}{3\Delta x^2}\left[(x-x_{j-1})^3\right]_{x_{j-1}}^{x_j}=\frac{2}{3\Delta x^2}\Delta x^3=\frac{2}{3}\Delta x \tag{A.7}$$

Integral A.3 for $i=j$ taking symmetry into account becomes:

$$-\int_{x_{j-1}}^{x_{j+1}}(\frac{d\phi_j}{dx})^2 dx=-\frac{1}{\Delta x^2}\int_{x_{j-1}}^{x_{j+1}}1dx \tag{A.8}$$

$$=-\frac{1}{\Delta x^2}\left[x\right]_{x_{j-1}}^{x_{j+1}}=-\frac{2}{\Delta x} \tag{A.9}$$

2. $|j-k|=1$ A.2 becomes:

$$\frac{1}{\Delta x^2}\int_{x_j}^{x_{j+1}}(x-x_j)(x_{j+1}-x)dx \tag{A.10}$$

$$=\left[\frac{1}{2}x^2x_{j+1}-\frac{1}{3}x^3-xx_{j+1}x_j+\frac{1}{2}x^2x_j\right]_{x_j}^{x_{x_j+1}}=\frac{1}{6\Delta x^2}\Delta x^3=\frac{1}{6}\Delta x \tag{A.11}$$

Finally A.3 has to be evaluated for this condition:

$$-\int_{x_j}^{x_{j+1}}\frac{d\phi_j}{dx}\frac{d\phi_{j+1}}{dx}dx=\frac{1}{\Delta x^2}\int_{x_j}^{x_{j+1}}1dx=\frac{1}{\Delta x^2}\left[x\right]_{x_j}^{x_{j+1}}=\frac{1}{\Delta x} \tag{A.12}$$

A.2. Proof that matrix M is invertible

Let M_n be a matrix with $M_n \in \mathbb{R}^{n\times n}$ given by:

$$m_{ij} = \begin{cases} a, & k = j \\ b, & |k-j| = 1 \\ 0, & otherwise \end{cases} \tag{A.13}$$

It's determinant is given by the Laplace expansion:

$$det(M_n) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} N_{ij} \quad \forall i \tag{A.14}$$

N_{ij} is the determinant of the matrix M' that is obtained by removing the i^{th} row and j^{th} column of M_n. This expression can be simplified using the definition of m_{ij}:

$$det(M_n) = aN_{11} - bN_{12} \tag{A.15}$$

N_{11} is equivalent to $det(M_{n-1})$, since the indices of rows and columns of M' are in consecutive order and M' is a $n-1 \times n-1$ matrix :

$$N_{11} = det(M') \tag{A.16}$$

$$M' = \begin{bmatrix} m_{22} & \dots & m_{2n} \\ \vdots & \ddots & \vdots \\ m_{n2} & \dots & m_{nn} \end{bmatrix} \tag{A.17}$$

N_{12} can be obtained by calculating the determinant of M' using the Laplace expansion:

$$M' = \begin{bmatrix} m_{21} & m_{23} & \dots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n3} & \dots & m_{nn} \end{bmatrix} \tag{A.18}$$

$$det(M') = m_{21} \cdot det(M'') \tag{A.19}$$

$$M'' = \begin{bmatrix} m_{33} & \dots & m_{3n} \\ \vdots & \ddots & \vdots \\ m_{n3} & \dots & m_{nn} \end{bmatrix} \tag{A.20}$$

In A.19 only the stated term has to be evaluated since all entries of the first column of the second sub matrix are zero. Therefore the determinant is zero. The row and column indices of M'' are in consecutive order and it is a $n-2 \times n-2$ matrix. Therefore M'' is equivalent to M_{n-2}. A.15 becomes:

$$det(M_n) = a \cdot det(M_{n-1}) - b^2 \cdot det(M_{n-2}) \tag{A.21}$$

Furthermore this implies $det(M_0) = 1$:

$$det(M_2) = a^2 - b^2 = a \cdot det(M_1) - b^2 \cdot 1 \tag{A.22}$$

$$\Rightarrow det(M_0) = 1 \tag{A.23}$$

Using the definition of 2.34 and 2.2.1 this can be seen as the following sequence:

$$a_0 = 1,\ a_1 = \frac{2\Delta x}{3} \tag{A.24}$$

$$a_{n+1} = \frac{2\Delta x}{3} \cdot a_n - \frac{\Delta x^2}{36} \cdot a_{n-1} \tag{A.25}$$

As described here [38] a recursive sequence converges if it is monotone and has a limit. A proof by induction shows that this sequence is monotone for $n \geq 1$.

Base case:

$$a_2 = (\frac{2\Delta x}{3})^2 - \frac{\Delta x^2}{36} = \Delta x^2(\frac{4}{9} - \frac{1}{36}) < \frac{2\Delta x}{3} = a_1 \tag{A.26}$$

Induction step: Assuming that $a_k < a_{k-1}$ holds, $a_{k+1} < a_k$ also holds:

$$a_{k+1} = \frac{2\Delta x}{3} \cdot a_k - \frac{\Delta x^2}{36} \cdot a_{k-1} < \frac{2\Delta x}{3} \cdot a_{k-1} - \frac{\Delta x^2}{36} \cdot a_{k-2} = a_k \tag{A.27}$$

The limit of this sequence is as follow:

$$\alpha = \lim_{n\to\infty} a_{n+1} = \lim_{n\to\infty} \Delta x \cdot \frac{2}{3} \cdot \lim_{n\to\infty} a_n - \lim_{n\to\infty} \Delta x^2 \cdot \frac{1}{36} \cdot \lim_{n\to\infty} a_{n-1} = 0 \cdot \alpha - 0 \cdot \alpha = 0 \tag{A.28}$$

Since this series is monotone and converges to zero as n goes to infinity, there is no $n \in \mathbb{N}$ for which $a_n = 0$. Therefore the determinant of the matrix M defined in 2.34 is not zero and M is invertible.

A.3. Equivalence of picewise linear polynomials and linear interpolation

A picewise linear polynomial in the form of:

$$u(x,t) = \sum_{j=1}^{N} c_j(t)\phi_j(x) \tag{A.29}$$

With ϕ_j being defined as A.1 and $c_j : \mathbb{R} \to \mathbb{R}$ is equivalent to linear interpolation with respect to x:

$$\hat{u}(x,t) = u_j + \frac{(u_{j+1} - u_j)(x - x_j)}{x_{j+1} - x_j} \tag{A.30}$$

for $x_j < x < x_{j+1}$ [39]. This can be shown by evaluating $u(x,t)$ between two neigbouring ϕ and $x_j < x < x_{j+1}$:

$$u(x,t) = \phi_j(x)c_j(t) + \phi_{j+1}(x)c_{j+1}(t) \tag{A.31}$$

$$= \frac{x_{j+1} - x}{\Delta x}c_j(t) + \frac{x - x_j}{\Delta x}c_{j+1}(t) \tag{A.32}$$

$$= \frac{x_{j+1} - x}{\Delta x}u_j + \frac{x - x_j}{\Delta x}u_{j+1} \tag{A.33}$$

$$= \frac{(x_{j+1}u_j - xu_j) + (xu_{j+1} - x_ju_{j+1})}{\Delta x} \tag{A.34}$$

$$= \frac{(u_{j+1} - u_j)x + x_{j+1}u_j - x_ju_{j+1}}{\Delta x} \tag{A.35}$$

$$= \frac{(u_{j+1} - u_j)x + x_ju_j + \Delta xu_j - x_ju_{j+1}}{\Delta x} \tag{A.36}$$

$$= u_j + \frac{(u_{j+1} - u_j)x - x_j(u_{j+1} - u_j)}{\Delta x} \tag{A.37}$$

$$= u_j + \frac{(u_{j+1} - u_j)(x - x_j)}{x_{j+1} - x_j} \tag{A.38}$$